UNIVERSIDAD POLITÉCNICA DE VALENCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA,
CARTOGRÁFICA Y TOPOGRÁFICA

Sistema de navegación para robots móviles basado en un ordenador de placa simple

Trabajo Fin De Máster

Autor: David Montero García

Tutores: Ángel Marqués Matéu

José Á. Garrido Sarasol

Septiembre, 2016

CONTENIDO

RESUMEN .....
1. INTRODUCCIÓN 1
1.1 Introducción 1
1.2 Motivación del trabajo 2
1.3 Objetivo del trabajo 3
1.4 Estructura del trabajo 3
2. ANTECEDENTES 5
2.1 Los SBC 5
2.2 TECNOLOGÍAS DE POSICIONAMIENTO 8
2.2.1 GNSS y SBAS 8
2.2.2 A-GNSS 10
2.2.3 Odometría y Accionamiento Diferencial Cinemático 10
2.3 Algoritmo Persecución 16
3. LENGUAJES DE PROGRAMACIÓN PARA LA RASPBERRY PI 20
Ruby 20
Lenguaje C 21
Visual Basic 21
Python 22
4. CONFIGURACIÓN DE LA RASPBERRY 24
5. VEHÍCULOS 39
ROVER 1 40
ROVER 2 47
ROVER 1 VS ROVER 2 57
6. SOFTWARE DESARROLLADO 59
7. PRUEBAS 65
7.1 Pruebas GNSS estático 65
7.2 Pruebas itinerarios 70
8. MEJORAS 77
9. CONCLUSIONES 78
BIBLIOGRAFÍA 79

INDICE DE FIGURAS

Figura 1: Single Board Computer. 1
Figura 2: Esquema robot cinemático diferencial. 11
Figura 3: Accionamiento diferencial en un robot. Fuente: Martin Lundgren. 12
Figura 4: Aproximación a punto mediante Persecución Pura. 17
Figura 5: Ruby. 20
Figura 6: C. 21
Figura 7: Visual Basic. 22
Figura 8: Python. 23
Figura 9: Raspberry Pi 2 Model B. 24
Figura 10: Puertos de la Raspberry Pi 2. 26
Figura 11: Escritorio Raspberry Pi 2 raspbian-jessie 27
Figura 12: Adaptador inalámbrico TP-LINK WN 823N. 28
Figura 13: Red WIFI creada con Raspberry 30
Figura 14: Logo escritorio remoto Remmina 32
Figura 15: Configuración conexión por VNC a Raspberry mediante Remmina 32
Figura 16: Modulo GNSS V2 para Raspberry 34
Figura 17: Especificaciones GNSS Ublox familia NEO-6. 35
Figura 18: Opciones avanzadas del menú de configuración Raspbian. 36
Figura 19: Interfaz del programa GPSMON en terminal en Raspberry. 38
Figura 20: Powerbank 8000mah marca Celly. 39
Figura 21: ROVER 1. 40
Figura 22: ROVER 1, vista de perfil. 41
Figura 23: ROVER 1, vista trasera. 41
Figura 24: Controlador de motores dual L298N. 42
Figura 25: Esquema conexiones controlador motores L298N. Fuente: www.luisllamas.es. 43
Figura 26: Pines empleados para conectar el controlador de motores. 44
Figura 27: Motores del ROVER 1. 45
Figura 28: Instalación de los motores del eje del ROVER 1. 46
Figura 29: ROVER 2. 47
Figura 30: ROVER 2, vista lateral. 48
Figura 31: Kit de locomoción RD02. Fuente: www.superrobotica.com 49
Figura 32: Kit locomoción y baterías ROVER 2. 50
Figura 33: Esquema Controlador MD25. Fuente: www.superrobotica.com 52
Figura 34: Configuración pull-up SDA y SCL de la Rapberry. 57
Figura 35: Movimientos del ROVER 1 59
Figura 36: Esquema navegación ROVER 1. 60
Figura 37: Esquema funcionamiento ROVER 2. 62
Figura 38: Zona pruebas GNSS estatico. Vista general. 66
Figura 39: Zona pruebas GNSS estático. Vista detalle. Fuente: Visor SIGNA IGN 66
Figura 40: Posiciones registradas en la primera observación. 67
Figura 41: Posiciones registradas en la segunda observación. 69

Figura 42: Zona de Pruebas, Recinto Ferial. Fuente: Visor IBERPIX.....70
Figura 43: Trayectoria. Fuente: Visor IBERPIX.....71
Figura 44: Itinerario 1 ROVER 2.....72
Figura 45: Itinerario 2 ROVER 2.....75
Figura 46: Itinerario 2 ROVER 2.....76

INDICE DE TABLAS

Tabla 1: Comparativa de dispositivos SBC.....6
Tabla 2: Especificaciones de RPI GPS ADD-ON V2.0 33
Tabla 3: Registros Controlador MD25. Fuente: www.superrobotica.com.....54
Tabla 4: Comandos del controlador MD25. Fuente: www.superrobotica.com.....55
Tabla 5: Tabla de resultados del itinerario del ROVER 2 73

RESUMEN

Los ordenadores de placa simple o simple board computers (SBC) constituyen una plataforma de reducido tamaño que puede ser incorporada a robots móviles. Los SBC incluyen un sistema operativo totalmente funcional junto con interfaces para conectar distintos dispositivos externos. En este proyecto se creará un sistema sencillo que incluye un SBC y un receptor GNSS. El proyecto incluye el desarrollo de un vehículo y un programa en Python encargado de la navegación del vehículo de manera autónoma siguiendo un itinerario. La plataforma de desarrollo será la conocida placa Raspberry Pi.

1. INTRODUCCIÓN

1.1 Introducción

Un dispositivo SBC (Single Board Computer) u ordenador de placa reducida, es un ordenador completo en un sólo circuito, el cual dispone de todas las características de un ordenador funcional en una sola tarjeta de tamaño reducido. En definitiva, alberga todo lo que necesita para su correcto funcionamiento en la placa base. A lo largo del presente trabajo, se realizará una definición más completa de estos dispositivos.

Actualmente, estos dispositivos se encuentran en auge debido principalmente a la buena relación de prestaciones-precio y a su reducido precio.

En los últimos años, los SBC o mini PCs han sufrido una gran evolución y desarrollo, aumentando su rendimiento y características, hasta el punto de ser utilizados como un ordenador al uso.

Hoy en día, los SBCs se encuentran en una multitud de dispositivos tanto industriales como domésticos. Para el ámbito doméstico, los ordenadores de placa simple de bajo coste tienen un mayor interés. Un ejemplo de estos es la Raspberry Pi.

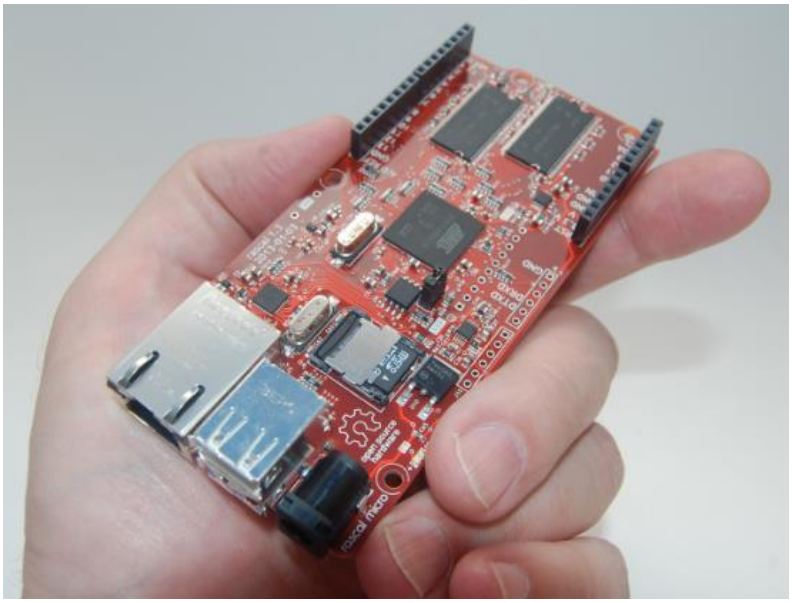

Figura 1: Single Board Computer.

Actualmente, los ordenadores simples de bajo coste están sufriendo un rápido y gran desarrollo, incluyendo mejores características técnicas, lo que permite utilizarlos en un sinfín de aplicaciones para robótica, domótica, IoT, etc. Además, el precio reducido de estos dispositivos permite tanto que el acceso a ellos sea multitudinario como el inicio en este campo de una gran parte de la población.

Por norma general, los SBC cuentan con las siguientes conexiones:

- ✓ Cámara.
- ✓ Bluetooth.
- ✓ Pantalla táctil.
- ✓ Antena GNSS.
- ✓ Conexión a Internet.
- ✓ Sensores inerciales.

Todas estas características permiten al dispositivo interactuar con el mundo y estar conectado en todo momento.

En el presente trabajo se describe la creación de un robot móvil auto guiado mediante sistemas de navegación por satélite (GNSS) con el fin último de desarrollar un vehículo autónomo.

Este robot móvil será controlado mediante el ordenador de placa simple conocido como Raspberry Pi. Además, la programación de éste se realizará en el lenguaje de programación Python.

1.2 Motivación del trabajo

La idea de realizar un proyecto de robótica aplicado a la Geomática surge a comienzos del presente año. Desde hacía un año sentía curiosidad por la programación en la Raspberry Pi. Me disponía a seleccionar un título para el trabajo de final de máster, cuando surgió la oportunidad de realizar un proyecto de dichas características enfocado a la Geomática.

Salvando el inconveniente de mis escasos conocimientos en electrónica y electricidad realizando un curso online de fundamentos de la electricidad, le propuse a mi tutor Ángel la idea, quien me animo a llevarla a cabo.

Ante la inexistencia dentro de la escuela de un trabajo semejante considere una gran oportunidad el desarrollo del mismo.

Con este trabajo se ha intentado aprovechar los conocimientos adquiridos a lo largo de la titulación aplicándolos a las nuevas tecnologías emergentes, como es el caso de los vehículos autónomos.

1.3 Objetivo del trabajo

El objetivo del trabajo es el desarrollo de un robot móvil, concretamente un vehículo terrestre, capaz de recorrer un itinerario de forma autónoma por medio del uso de sistemas GNSS y llegando a una versión estable y funcional.

1.4 Estructura del trabajo

Para llevar a cabo el objetivo principal del presente trabajo, se plantean los objetivos secundarios que se detallan a continuación y que dotaran de la siguiente estructura al trabajo:

Antecedentes.

En este apartado se realizará:

- ✓ Una introducción a los ordenadores de placa simple o reducida (SBCs) de bajo coste. Para ello se hará una breve comparativa de los diversos tipos de dispositivos SBC existentes en la actualidad utilizados para el desarrollo de proyectos básicos de electrónica, y la posterior elección de uno de ellos.

- ✓ Una mención a las tecnologías de posicionamiento disponibles, de las cuales se escogerán las óptimas para su empleo en la aplicación.

- ✓ Una breve explicación de los conceptos y algoritmos empleados en la navegación y guiado del robot.

Configuración de la Raspberry.

En este punto se definirá el procedimiento de configuración llevado a cabo en la Raspberry para su posterior integración en el vehículo.

Vehículos

Una vez configurada la Raspberry, será incorporada al vehículo. En este apartado se presentarán los prototipos diseñados y empleados.

Pruebas

En este apartado se realizarán diferentes pruebas con los vehículos utilizados en diferentes itinerarios, con la finalidad de comprobar el nivel de precisión de éstos al recorrer diversos trayectos.

Mejoras

En este punto se propondrán posibles mejoras a realizar en los vehículos.

Conclusiones

Finalmente, se expondrán las conclusiones extraídas de la realización del presente trabajo

2. ANTECEDENTES

2.1 Los SBC

Un ordenador de placa reducida o Single Board Computer (SBC) se puede definir como un PC completo en un sólo circuito. En él se encuentra el microprocesador y la memoria RAM, así como las demás conexiones y puertos para su correcto funcionamiento.

Al contrario que en los ordenadores convencionales, en los SBC los componentes como la memoria RAM y el microprocesador no se pueden remplazar debido a que están integrados en la placa, por lo que es imposible actualizar estos componentes con el paso del tiempo. Por tanto, y como consecuencia de la imposibilidad de cambiar los componentes de la placa, se debe prestar especial atención a la elección del SBC que vamos a utilizar.

Las nuevas características que los fabricantes de estos dispositivos están incorporando actualmente son:

- Salidas de audio.
- Salidas de video.
- Red.
- Puertos entrada/salida en un tamaño más reducido.
- Etc.

Además, cabe decir que los ordenadores de placa reducida o Single Board Computer, tienen un consumo energético inferior al de los ordenadores convencionales.

En la Tabla 1, se muestra un resumen con los ordenadores de placa reducida actuales más populares y sus características.

Tabla 1: Comparativa de dispositivos SBC

				
Nombre	Raspberry Pi 2	Banana Pi M2	BeagleBone Black	Orange Pi One
Procesador	Broadcom BCM2836 ARM Cortex-A7 4 núcleos @ 900 MHz	A31S ARM Cortex-A7 4 núcleos	AM335x 1GHz ARM Cortex-A8	H3 Quad-core Cortex-A7 H.265/HEVC 4K
RAM	RAM 1 GB LPDDR2 SDRAM 450 MHz	1GB DDR3	512MB DDR3	512MB DDR3
GPU	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0	PowerVR SGX544MP2 con soporte a OpenGL ES 2.0 y OpenCL 1.x	Power VR SGX530	Mali400MP2 GPU @600MHz Soporte para OpenGL ES 2.0
Almacenamiento	Tarjeta Micro SD	MicroSD Card(hasta 64GB)	Memoria interna de 2GB eMMC. Ampliable por tarjeta micro SD	MicroSD Card(hasta 64GB)
Puertos	4 x USB 2.0 40 GPIO pin HDMI 1.4 Ethernet 10/100 Mbps 1 x Combo audio/micro Interfaz de cámara (CSI) Interfaz de Pantalla (DSI)	4x USB 2.0 40 GPIO pin Ethernet RJ45 10/100/1000 WiFi 802.11b/g/n HDMI,LVDS/RGB 3.5 mm Jack Micrófono Conector de cámara	Mini HDMI Ethernet 10/100Mbps RJ-45 Jack 2 USB 2.0 2 x 46 E/S de tipo: 65 digitales, 7 analógicas, 4 serie, 2 SPI, 2 I2C, 8 PWM, 4	1 USB 2.0 40 GPIO pin 1 USB 2.0 OTG Ethernet RJ45 10/100M Interfaz de cámara (CSI)
Dimensiones	85,60x56,5 mm	92mm x 60mm	86 x 53 mm	69 mm × 48mm
Peso	45 g			36g
Consumo	5v, 1A,	5V ,2A	5V , 0.5A	5V,1A
OS soportados	Android,Linux Windows 10 IoT	Android y Linux (Debian y Ubuntu)	Angstrom Linux Android	Android 4.4, Ubuntu, Debian,

			Ubuntu	Imágenes de Rasberry Pi
Precio	40 €	40€	50 €	15 €

En la anterior tabla comparativa, se puede observar que todos los SBC analizados disponen de unas características semejantes. Cabe reseñar que varios de ellos cuentan con componentes adicionales, como por ejemplo Wi-Fi y bluetooth integrados.

Las características comunes en todos los ordenadores de placa reducida anteriormente mencionados son:

- Se pueden instalar sistemas operativos basados en Linux.
- Disponen de salidas para video y audio.
- Disponen de pines digitales y analógicos de entrada y salida.
- Tienen conexión a Internet
- Cuentan con puertos de datos USB.

Tras analizar la tabla anterior, concluyendo que los SBC analizados son prácticamente similares, cabe reseñar que de las placas anteriormente mencionadas, la más conocida y extendida entre la población es la Raspberry Pi, modelo 2. El hecho de que este SBC lo posea una gran comunidad de usuarios, es una ventaja comparativa primordial frente al resto de dispositivos analizados puesto que ello nos asegura una gran cantidad de información y una rápida resolución de los problemas que puedan plantearse a lo largo del desarrollo del presente trabajo.

Así pues, y por los motivos anteriormente expuestos, la placa elegida para el desarrollo de este trabajo es la Raspberry Pi 2.

2.2 TECNOLOGÍAS DE POSICIONAMIENTO

Como se ha mencionado anteriormente, el presente trabajo versa sobre la construcción y desarrollo de un robot móvil, concretamente un vehículo, capaz de recorrer un itinerario de forma autónoma. Para ello, es imprescindible la utilización de métodos de posicionamiento.

A continuación, se muestra una clasificación de los métodos de posicionamiento empleados para el posicionamiento y navegación de los vehículos diseñados.

2.2.1 GNSS y SBAS

El sistema global de navegación por satélite, o GNSS, es el acrónimo que se refiere al conjunto de tecnologías de sistemas de navegación por satélite que ofrecen un posicionamiento geoespacial con cobertura global de manera autónoma y/o con sistemas de aumentación.

El principio de funcionamiento de los sistemas globales de navegación por satélite se basa en una constelación de satélites artificiales que orbitan a una altura de 20.000 kilómetros sobre la tierra, emitiendo ondas electromagnéticas desde el espacio. Estas ondas son recibidas por el dispositivo, en nuestro caso el sensor GNSS acoplado a la Raspberry. La posición de los satélites o efemérides es conocida, siendo transmitida por los satélites en las señales emitidas.

Es necesario recibir la señal de al menos cuatro satélites para el cálculo de la posición, debido a que son cuatro las incógnitas a resolver. Estas son las coordenadas en el espacio del punto a calcular, en coordenadas cartesianas (X,Y,Z) o latitud, longitud y altura en coordenadas geográficas y el tiempo [1]

Se requiere de sistemas de medición de tiempos muy precisos, para ello se utilizan relojes atómicos de alta precisión en los satélites con una estabilidad del orden de 10^{-13} - 10^{-14} segundos/día. Los receptores suelen ir incorporados de relojes de cuarzo, de menor estabilidad.

El cálculo de la posición se lleva a cabo mediante una trilateración, conocidas las coordenadas de los satélites y la distancia entre el receptor y los satélites.

El cálculo de las distancias entre satélite y receptor, se basa en la propagación de las ondas electromagnéticas en el vacío, cuya velocidad es de 300.000 km/s.

Estas distancias se pueden calcular por medio de métodos conocidos como pseudodistancia por medición en código o por medición en diferencia de fase.

Medición en código.

El método de medición en código calcula la distancia por diferencia de tiempo como $d = v * t$, siendo d la distancia entre satélite y receptor la velocidad de la luz en el vacío, y t el tiempo que tarda la señal en recorrer el trayecto entre satélite y receptor.

Medición en fase.

El método de medición en fase se basa en la medición del desfase de la onda, se controla en fase la emisión de ésta con una frecuencia y posición conocidas. Al controlar la fase, se observa continuamente la evolución del desfase entre la señal recibida y la generada por el receptor. El observable es el desfase y éste cambia según lo hace la distancia satélite-antena receptora.
Correlacionando continuamente ambas portadoras a partir del momento de conexión con el satélite, el receptor podrá determinar la cantidad de ciclos enteros debido a los cambios de distancia entre satélite y receptor y medir la fracción de ciclo entre ambas señales (ϕ).
La diferencia entre ciclos observados y la cantidad total será la ambigüedad inicial N, que será invariable para toda la sesión siempre que no se produzcan cortes en la señal, ya que en ese momento se perderá la cuenta de ciclos enteros y aparecerá una nueva ambigüedad.
El problema que se plantea en la medida de fase es la dificultad que implica la obtención del número inicial de ciclos enteros en la portadora (N) contenidos en la distancia D que hay entre satélite y receptor.

Los sensores GNSS de bajo coste actuales utilizan mediciones de código para el cálculo de la posición.

Los principales errores del sistema GNSS son los errores en la medida de pseudodistancia, por errores en la ionosfera y la troposfera.

Los sistemas SBAS o sistemas basados en aumentaciones, modelizan estos errores y los envían a través de satélites geoestacionarios reduciendo así considerablemente el error en el cálculo de la posición.

En la actualidad, los receptores GNSS disponibles para SBC ya disponen de la capacidad de uso de los sistemas SBAS.

2.2.2 A-GNSS

A-GNSS, GNSS asistido o A-GPS es una técnica que permite reducir los tiempos de inicialización del GPS. Utilizando una conexión WI-FI, se descargan a través de Internet las efemérides de los satélites. Conociendo la posición aproximada de los satélites se consigue minimizar el tiempo a la hora de fijar la posición.

La precisión de los receptores GNSS con solución de código, junto con los Sistemas SBAS, nos proporciona una precisión en posicionamiento con condiciones óptimas en torno a los 2m.

Debido a la necesidad de una alta precisión en el posicionamiento del vehículo en este trabajo, se ha considerado el uso de la odometría para el posicionamiento del robot en cortas distancias.

2.2.3 Odometría y Accionamiento Diferencial Cinemático

La navegación a estima es un método utilizado para determinar la posición actual o instantánea de un vehículo, a partir de la posición anterior y conociendo el rumbo y la velocidad de avance en un periodo de tiempo determinado. Es un proceso de estimación de la posición de un vehículo, basado únicamente en velocidad, la dirección y el tiempo transcurrido desde la última posición conocida.

Actualmente, la mayor parte de los vehículos de tierra robóticos se basan en la navegación a estima como marco de sus sistemas de navegación. Una forma sencilla de implementar la navegación a estima se denomina odometría.

La Odometría es un método para proporcionar información sobre el desplazamiento del vehículo en base a la rotación de sus motores o ruedas. La rotación se mide en la rueda o en los encoders. Los encoders son sensores que miden las revoluciones o el número de rotaciones de los ejes de las ruedas, proporcionando datos que combina con las ecuaciones de cinemática directa pueden ser traducidos a la información relativa a los cambios en la posición y orientación del vehículo.

Un robot móvil de cinemática diferencial es aquel que cuenta con dos ruedas motrices unidas por un eje (Figura 2). Cada una de estas ruedas tiene su propio motor que la hace girar en el plano perpendicular al eje que une ambas ruedas.

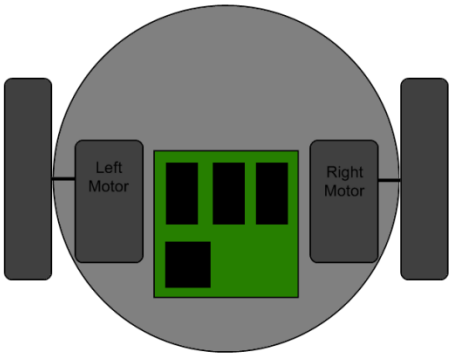

Figura 2: Esquema robot cinemático diferencial.

La teoría que respalda la cinemática de accionamiento diferencial es bastante sencilla: Cada rueda del robot móvil, en un estado de movimiento, siempre debe girar alrededor de un punto que se encuentra en algún lugar del eje común de las dos ruedas. Este punto se llama a menudo el Centro Instantáneo de Rotación (CIR) o centro instantáneo de curvatura (ICC). Un robot de accionamiento diferencial, cambia la posición del CIR simplemente variando las velocidades de las dos ruedas. Dos motores de accionamiento separados, uno para cada rueda, proporcionan un control de velocidad independiente a las ruedas izquierda y derecha. Es esta propiedad la que dota al robot de la posibilidad de tomar diferentes trayectorias y caminos.

En la Figura 3 se muestra una el esquema de un robot móvil diferencial:

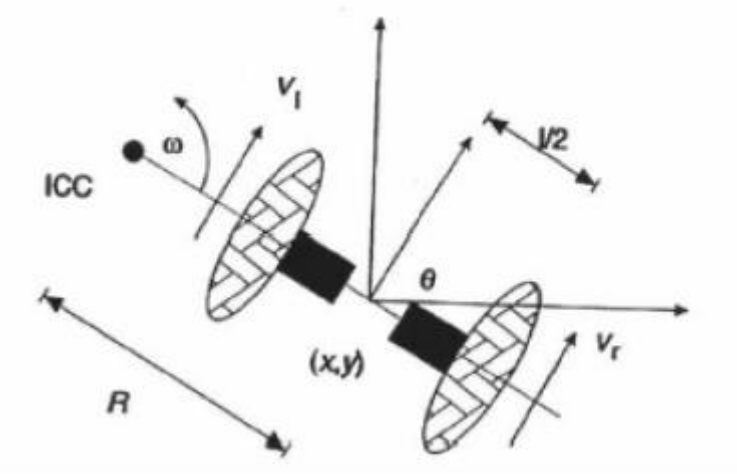

Figura 3: Accionamiento diferencial en un robot. Fuente: Martin Lundgren.

Como se observa en la Figura 3, cada rueda sigue una trayectoria que se mueve alrededor del CIR o ICC en la misma velocidad angular ω, por lo tanto:

$$\omega\left(\mathrm{R}+\frac{\mathrm{I}}{2}\right)=v_r$$

$$\omega\left(\mathrm{R}-\frac{\mathrm{I}}{2}\right)=v_l$$

Donde l es la distancia entre las dos ruedas, la rueda derecha se mueve con velocidad y v_r la rueda izquierda se mueve con velocidad v_l. R es la distancia del CIR en el punto medio entre las ruedas.

Todos estos parámetros de control son funciones del tiempo, por tanto:

$$R=\frac{I}{2}*(v_l+v_r)/(v_r-v_l)$$

$$\omega = (v_r - v_l)/\mathrm{l}$$

Existen dos casos especiales que provienen de estas ecuaciones:

- Si v_l = v_r, y por tanto el radio R es Infinito. El robot se mueve en una línea recta.
- Si v_l = $-v_r$, y por tanto el radio es cero. El robot gira en sobre sí mismo.

Las ecuaciones de cinemática directa se pueden derivar fácilmente ahora que hemos establecido lo esencial. Nuestra atención se centra en las coordenadas X e Y, junto con el cambio de orientación con respecto al tiempo y:

- Siendo θ el ángulo de orientación en radianes, medido con respecto al eje de las x en sentido anti horario o levógiro.
- Siendo m(t) y θ(t) la velocidad y la orientación en función del tiempo para el robot.

Por tanto, la solución será la siguiente:

$$\mathrm{dx/dt} = \mathrm{m(t)cos(\theta(t))} \qquad 3.1$$

$$\mathrm{dy/dt} = \mathrm{m(t)sin(\theta(t))} \qquad 3.2$$

El cambio de orientación con respecto al tiempo es igual a la velocidad angular ω. Por lo tanto:

$$\mathrm{d\theta/dt} = \omega = (v_r - v_l)/\mathrm{l} \quad 3.3$$

La integración de esta ecuación da una función para la orientación de robots con respecto al tiempo.

La orientación inicial θ (0) también se sustituye por θ_0:

$$\theta(t) = (v_r - v v_l)t/l + \theta_0 \qquad 3.4$$

Puesto que la velocidad en las funciones de (3.1) y (3.2) anterior simplemente es igual a la velocidad media de la dos ruedas, que es m(t)=(vr+vl)/2, de la integración de esta en (3.1) y (3.2) se obtiene:

$$dx/dt = [(v_r + v_l)/2]\cos(\theta(t)) \qquad 3.5$$

$$dy/dt = [(v_r + v_l)/2]\sin(\theta(t)) \qquad 3.6$$

El paso final es la integración de las ecuaciones (3.5) y (3.6), tomando las posiciones iniciales x (0) = x_0 e Y (0) = y_0 para obtener:

$$x(t) = x_0 + l/2(v_r + v_l)/(v_r - v_l)[sin((v_r - v_l)t/l + \theta_0) - sin(\theta_0)] \quad 3.8$$

$$y(t) = y_0 - l/2(v_r + v_l)/(v_r - v_l)[cos((v_r - v_l)t/l + \theta_0) - cos(\theta_0)] \quad 3.9$$

Atendiendo que l/2(vr + vl)/(vr - vl) = R y que (vr - vl)/l = ω para la configuración diferencial, las ecuaciones (3.8) y (3.9) se pueden reducir a:

$$x(t) = x_0 + R[sin(\omega t + \theta_0) - sin(\theta_0)] \qquad 4.0$$

$$y(t) = y_0 - R[cos(\omega t + \theta_0) - cos(\theta_0)] \qquad 4.1$$

Esta es la teoría básica sobre la navegación a estima en un robot móvil con ruedas usando la acción o dirección diferencial. Lo único que uno tiene que hacer es sustituir los términos v_l y v_r con S_R y S_L, para utilizar desplazamientos en lugar de velocidades. Los términos S_R y S_L son las distancias recorridas por la rueda derecha y la izquierda respectivamente.

Las ecuaciones resultantes, a partir de las expresiones (3.8) y (3.9), se convierten en:

$$x(t) = x_0 + l/2(S_R + S_L)/(S_R - S_L)[sin((S_R - S_L)/l + \theta_0) - sin(\theta_0)] \quad 4.2$$

$$y(t) = y_0 - l/2(S_R + S_L)/(S_R - S_L)[cos((S_R - S_L)/l + \theta_0) - cos(\theta_0)] \quad 4.3$$

INCONVENIENTES DE LA ODOMETRÍA

La estimación por odometría nos proporciona una estimación de la posición de nuestro vehículo, pero esta técnica no está libre de errores, y a continuación se muestran las principales fuentes de error, clasificados como errores sistemáticos y aleatorios.

Errores sistemáticos

Los errores sistemáticos más relevantes son:

- El radio de las ruedas no es exactamente el radio medido, debido a que las ruedas no son rígidas y sufren deformaciones.

- Otro error vendría introducido por la resolución del encoder, cuanto mayor sea ésta, menor error tendremos.

Errores aleatorios

El error aleatorio más relevante es, en el caso de que la rueda patine, que se producirá desplazamiento angular pero no lineal, que es en el que se basa el método. En ese momento, nuestras ecuaciones fallarán debido a que asumirán que el robot se está desplazando cuando no lo hace. Lo mismo puede pasar cuando el vehículo se encuentre con una irregularidad del terreno, como por ejemplo un bache.

Debido a esto podríamos afirmar que la odometría no es una manera fiable de localizar a nuestro robot. Cabe reseñar que es una forma sencilla de estimar la posición del robot, la cual va acumulando errores, tanto sistemáticos como no sistemáticos. Debido a esto, en trayectorias largas (más de un par de metros), la estimación de la posición por odometría empieza a acumular mucho error. Como consecuencia, tenemos que buscar la forma de aprovechar esta

información y corregir esa estimación, con el objetivo de tener un sistema de localización del robot que funcione correctamente.

Una vez resumidas brevemente las tecnologías para el posicionamiento y navegación en los robots móviles, y dado que para la navegación autónoma con un vehículo se requiere un posicionamiento con una cierta precisión, objeto del presente trabajo, la mejor solución para el posicionamiento es la solución combinada GNSS con odometria, puesto que es la solución que mejor precisión nos ofrece. Combinado ambas, conseguimos realizar itinerarios largos gracias al sensor GNSS y posición en cortas distancias gracias a la estimación por odometria.

Es recomendable hacer uso del A-GNSS en el caso de que nuestro receptor sea compatible, con el objetivo de reducir el tiempo de inicialización. Además, si el dispositivo dispone de la recepción de varias constelaciones de satélites GNSS (GPS-NAVSTAR y GLONASS), éste dispondrá de una mayor cantidad de observables para la obtención de la posición, y los tiempos de inicialización serán más bajos.

2.3 Algoritmo Persecución

A continuación se detalla la formulación empleada en el algoritmo de persecución pura o pure pursit, presentado en Donat (2015) a partir del cual se calcula la cantidad de movimiento a aplicar de cada uno de los motores necesaria para llegar a un punto.

El concepto del algoritmo de persecución pura es el de calcular la curvatura que llevará el vehículo desde su posición actual a un punto objetivo.

El punto objetivo se define como el punto que está a distancia D del el punto la posición actual del vehículo.

Posteriormente, se define un círculo y se pasa a través del punto objetivo y la posición actual del vehículo. Conviene, por último, garantizar un algoritmo de control que elija un ángulo de dirección en relación a este círculo. De hecho, el vehículo cambia continuamente la curvatura de los arcos que define con respecto el punto objetivo.

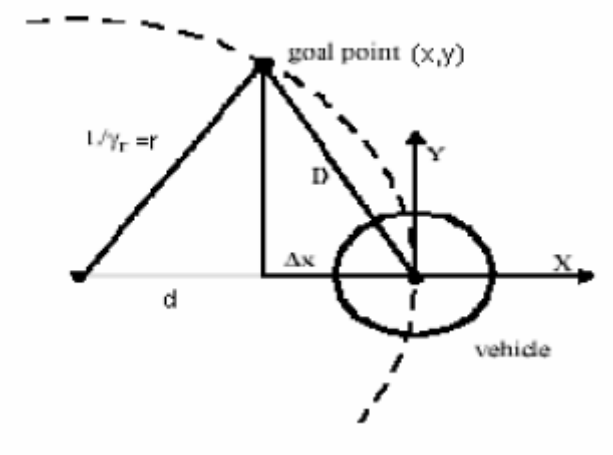

Figura 4: Aproximación a punto mediante Persecución Pura.

Es importante tener en cuenta que el desarrollo del algoritmo de persecución pura descrito se muestra en el sistema de coordenadas del vehículo. En el sistema de coordenadas del vehículo o body frame (b-frame) se define el eje, y en la dirección de avance del vehículo, el eje z está hacia abajo y el eje x perpendicular al eje y .Este tipo de sistemas son conocidos como sistemas right hand o mano derecha. Por lo tanto, todas las coordenadas utilizadas deben ser transformadas al body frame para que el algoritmo funcione correctamente.

En nuestro caso, las coordenadas objetivo o coordenadas de la ruta se encuentran en el sistema de referencia WGS 84 y en la proyección UTM (Universal Transversa de Mercator) zona 30. La posición del vehículo está definida a partir del sensor GNSS, y el sistema de referencia del vehículo se encuentra en sistema de referencia WGS 84 UTM.

Para hacer que ambos sistemas sean coincidentes, no es necesario trasladar las coordenadas objetivo y las posiciones GNSS al sistema de referencia del body frame o vehículo, ya es el mismo sistema de referencia, únicamente tenemos que pasar la posición de vehículo a la proyección UTM y el huso 30.

Siendo (x_r, y_r) la posición actual del robot, y (x_g, y_g) el punto objetivo en el sistema de coordenadas del vehículo (b-frame):

$$x_{gv} = (x_g - x_r)cos(\Phi) + (y_g - y_r)sin(\Phi)$$

$$y_{gv} = -(x_g - x_r)sin(\Phi) + (y_g - y_r)cos(\Phi)$$

Donde (xgv, ygv) es el punto objetivo en el sistema de coordenadas del vehículo y Φ es el rumbo del vehículo actual.

D se define como la distancia entre la posición actual del vehículo y el punto objetivo.

Δx es la variación en x del punto objetivo desde el origen, y 1/γr es el radio del círculo que pasa por el centro del vehículo y el punto objetivo. La curvatura que empleara el vehículo se calcula por medido de la siguiente expresión:

$$\gamma_r = 2\Delta x/D^2$$

La derivación de esta fórmula se basa a partir dos ecuaciones elementales:

1) $x^2 + y^2 = D^2$

2) $x + d = r$

(x, y) son las coordenadas del punto objetivo. La primera ecuación es el resultado de aplicar el teorema de Pitágoras en el triángulo rectángulo más pequeño en la Figura 4, y la segunda ecuación proviene de la suma de los segmentos de línea en el eje x.

$$d = r - x$$

$$(r - x)^2 + y^2 = r^2$$

$$r^2 - 2rx + x^2 + y^2 = r^2$$

$$2rx = D^2$$

$$r = D^2/2x$$

$$\gamma r = 2x/D^2$$

Este algoritmo ha sido implementado en el vehículo número 2 para que de manera autónoma, y a partir únicamente de la posición de los puntos por los que el robot debe visitar en el transcurso del itinerario, el vehículo calcule para cada instante la cantidad de movimiento que debe realizar para alcanzar cada punto del itinerario.

El único inconveniente que posee el método es que, para que coincidan los sistemas de coordenadas del vehículo y el sistema de coordenadas UTM, el vehículo debe ser orientado en una dirección determinada antes de iniciar el recorrido. En futuras mejoras, se prevé implementar una brújula para eliminar este inconveniente.

3. LENGUAJES DE PROGRAMACIÓN PARA LA RASPBERRY PI

A continuación, se muestran los principales lenguajes de programación utilizados para desarrollar proyectos en el dispositivo Raspberry Pi. Tras el análisis de todos ellos, se tomará la decisión de cuál es el óptimo para el desarrollo del presente trabajo.

Ruby

Ruby, presentado en 1995, es un lenguaje de programación interpretado, reflexivo y orientado a objetos.

Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos.

Se puede considerar como un lenguaje de programación interpretado. Cabe destacar que su implementación oficial es distribuida bajo una licencia de software libre.

Figura 5: Ruby.

Lenguaje C

C es un lenguaje de programación orientado a la implementación de Sistemas Operativos, concretamente Unix. Es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

La primera estandarización del lenguaje C se produjo con el estándar ANSI X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas

Figura 6: C.

Visual Basic

Visual Basic, desarrollado por Alan Cooper para Microsoft, es un lenguaje de programación dirigido por eventos. Este lenguaje de programación es un dialecto de BASIC, con importantes mejoras. Fue presentado en 1991.

La última versión fue la 6, liberada en 1998, para la que Microsoft extendió el soporte hasta marzo de 2008.

Visual Basic contiene un entorno de desarrollo integrado o IDE que integra editor de textos para edición del código fuente, un depurador, un compilador y un editor de interfaces gráficas o GUI.

Figura 7: Visual Basic.

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación estructurada. Es un lenguaje interpretado y multiplataforma.

Es administrado por la Python Software Foundation, y posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Figura 8: Python.

Tras el análisis de los lenguajes de programación anteriormente mencionados, se ha tomado la decisión de escoger para el desarrollo del presente trabajo el denominado Python. Se ha decidido utilizar este lenguaje de programación debido principalmente a:

- Dispone de las características necesarias para llevar a cabo el trabajo que vamos a desarrollar.

- Su lenguaje lo podemos considerar como familiar en el Máster en Geomática y Geoinformación, ya que ha sido el lenguaje de programación principal empleado durante toda la titulación.

4. CONFIGURACIÓN DE LA RASPBERRY

Para la correcta integración del SBC como cerebro de los robots, se han integrado diversos componentes y se ha realizado una serie de procedimientos, los cuales se detallan durante el presente apartado.

Raspberry Pi 2 Model B

De entre los diversos modelos de ordenadores de placa reducida de la marca Raspberry, el trabajo se ha llevado a cabo utilizando la Raspberry Pi 2 Model B, debido a que a fecha de comienzo de la realización de este trabajo, era el último modelo y el más eficiente.

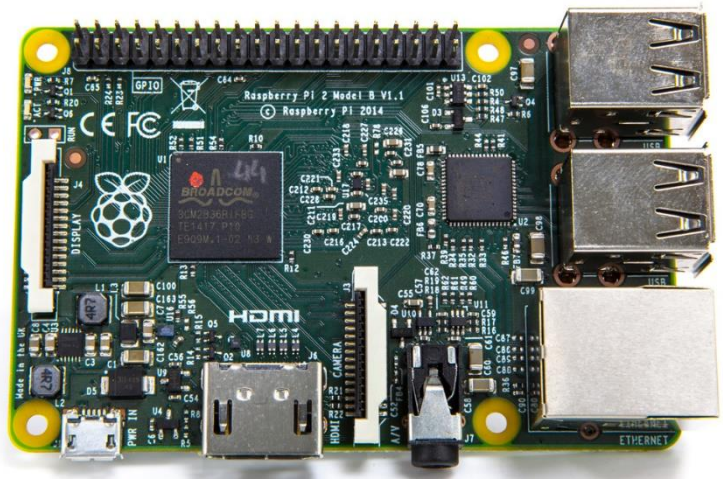

Figura 9: Raspberry Pi 2 Model B.

A continuación, se detallan las características de la Raspberry 2:

- Es aproximadamente 6 veces más rápida que los modelos anteriores en las tareas más frecuentes.

- Posee un procesador Broadcom BCM2836 ARM11 ARMv7 ARM Cortex-A7 de 4 núcleos @ 900 MHz y un procesador gráfico Broadcom VideoCore IV 250 MHz, lo que capacita a la Raspberry para reproducir video en Full HD (1920×1080 píxeles).

- Contiene una salida de video HDMI, cuatro puertos USB 2.0 y una ranura para tarjetas microSD, donde se almacena el sistema operativo.

- Cuenta con un puerto Ethernet RJ 45 a 10/100 Mbps de velocidad para posibilitar la conexión a Internet.

- Posee 40 pines GPIO (General Purpose Input/Output). Estos son pines de entrada/salida digitales, los cuales serán utilizados para la comunicación de la Raspberry con las controladoras de los motores de los vehículos.

- Sus dimensiones son de 85,60x56,5 mm.

- Su peso es de tan solo 45 g.

- El consumo eléctrico es de aproximadamente 5v, 900mA, aunque este depende de la carga de trabajo de los 4 cores y de los periféricos añadidos.

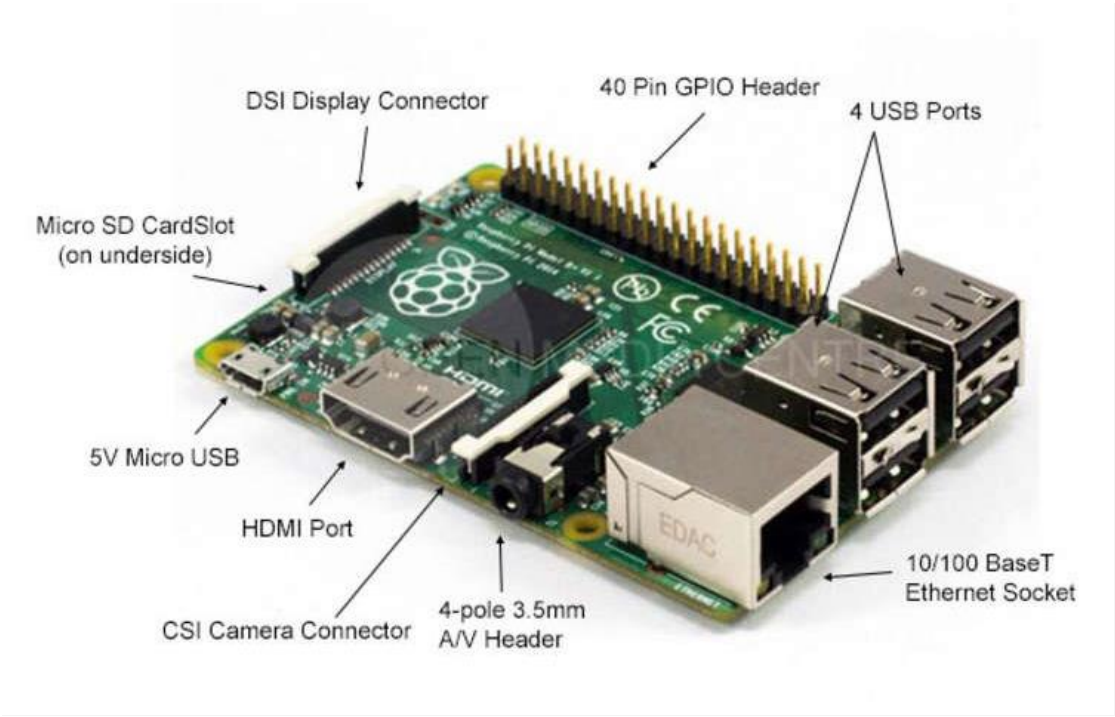

Figura 10: Puertos de la Raspberry Pi 2.

Una vez presentada la placa que se integrará en los vehículos, procedemos a analizar los periféricos añadidos y su configuración para la posterior integración en estos.

Sistema Operativo

De los muchos sistemas operativos disponibles en la página oficial del dispositivo para la Raspberry, se ha escogido el sistema operativo Raspbian, debido a que es el sistema operativo oficial soportado por la fundación Raspberry. Además, Raspbian incluye preinstalado una gran cantidad de software para educación y programación como Scratch, Sonic Pi, Java, Mathematica y Python, entre otros.

La imagen del sistema operativo Raspbian, empleada para la realización del trabajo, se corresponde con la versión de nombre raspbian-jessie.img, con fecha 09/02/2016.

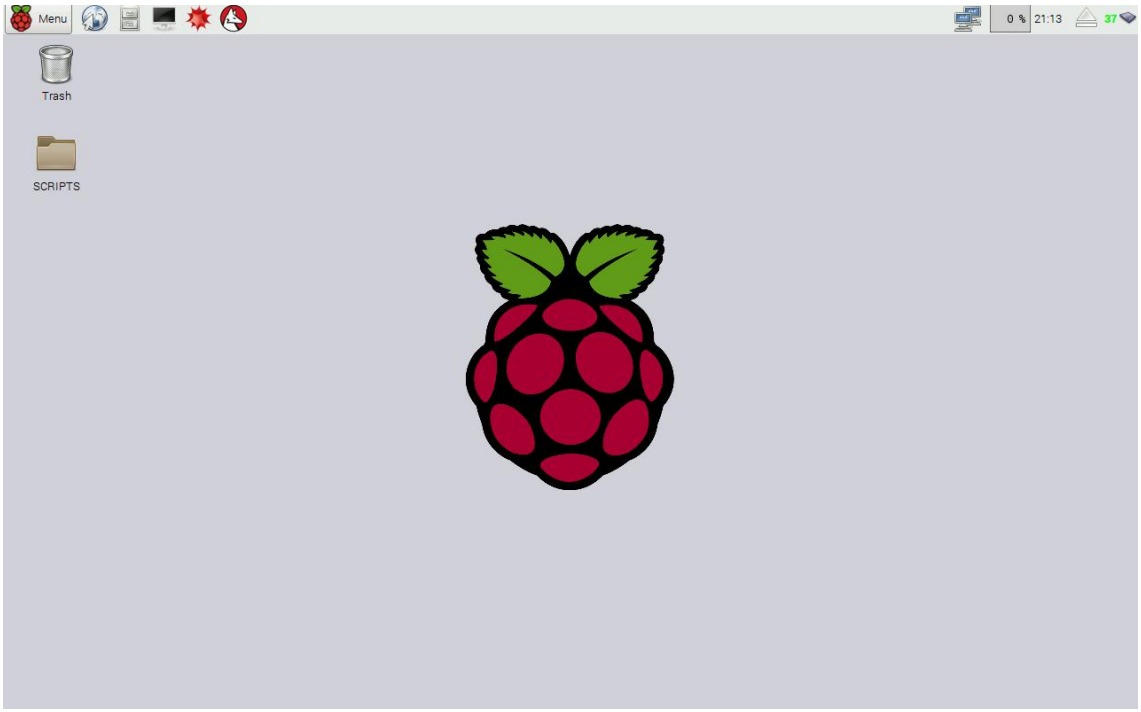

Figura 11: Escritorio Raspberry Pi 2 raspbian-jessie

Periféricos

Además, a la Raspberry Pi 2, se le ha añadido los siguientes periféricos:

- ✓ Un adaptador inalámbrico WI-FI USB.
- ✓ Un módulo GNSS.

El adaptador inalámbrico utilizado es el TP-LINK WN 823N v1 con las siguientes características:

CARACTERÍSTICAS DE HARDWARE

- Interfaz: USB 2.0
- Botones: Botón de Configuración Wi-Fi Protegida (WPS)
- Dimensiones: 1.54 x 0.72 x 0.31 in. (39 x 18.35 x 7.87mm)
- Tipo de Antena: Interna

CARACTERÍSTICAS INALÁMBRICAS

- Estándares Inalámbricos: IEEE 802.11b, IEEE 802.11g, IEEE 802.11n
- Frecuencia: 2.400~2.4835GHz
- Tasa de Señal:
  - 11b: Hasta 11Mbps (dinámica)
  - 11g: Hasta 54Mbps (dinámica)
  - 11n: Hasta 300Mbps (dinámica)
- Sensibilidad de Recepción
  - 300M: -65dBm@10% PER
  - 270M: -65dBm@10% PER
  - 130M: -68dBm@10% PER
  - 108M: -68dBm@10% PER
  - 54M: -72dBm@10% PER
  - 11M: -85dBm@8% PER
  - 6M: -87dBm@10% PER
  - 1M:-93dBm@8% PER
- Potencia de Transmisión: <20dBm (EIRP)
- Modos Inalámbrico:
  - Modo Soft AP
  - Modo Client (soporta Ad-hoc/red de infraestructura)
- Seguridad Inalámbrica:
  - 64/128-bit WEP
  - WPA-PSK / WPA2-PSK
  - WPA / WPA2
- Tecnología de Modulación: DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM

Figura 12: Adaptador inalámbrico TP-LINK WN 823N.

La principal característica del adaptador la cual lo hace indispensable para nuestro propósito, es que soporta el modo Inalambrico Ad-hoc. Esto nos permite crear con la Raspberry un punto de acceso inalámbrico, o dicho de otra manera, convertir a la Raspberry en un router al cual podamos conectarnos desde otro ordenador y comunicarnos con ella, eliminando así la necesidad de tener que conectar monitor, teclado y ratón a la Raspberry. En definitiva, nos permite controlar la Raspberry de forma remota.

Configuración

Para configurar la Rapberry como un punto de acceso inalámbrico debemos modificar, con el editor de textos nano, el fichero interfaces ubicado en la carpeta network, escribiendo el siguiente comando en el terminal:

```
sudo nano /etc/network/interfaces
```

De tal forma que el fichero quede con el siguiente contenido.

```
auto lo

iface lo inet loopback

iface eth0 inet manual

auto wlan0

iface wlan0 inet static

address 192.168.1.1

netmask 255.255.255.0

wireless-channel 10

wireless-essid RPIWireless

wireless-mode ad-hoc


allow-hotplug wlan1

iface wlan1 inet manual

wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Tras reiniciar la Raspberry, el adaptador inalámbrico crea una red con una dirección IP estática de nombre RPIWireless y sin clave, a la cual podremos conectarnos.

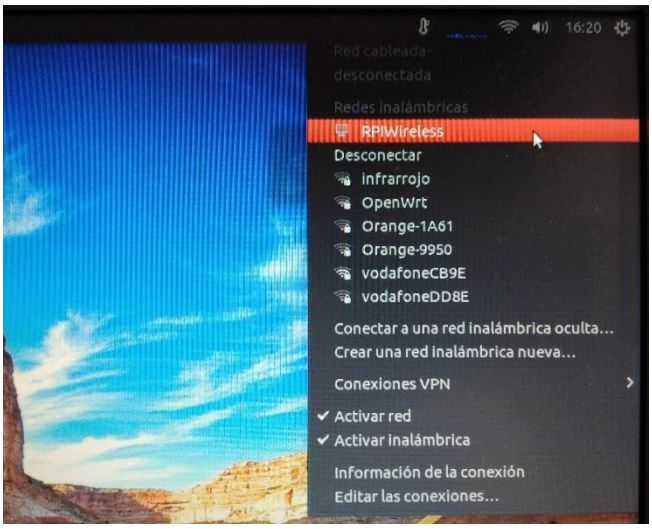

Figura 13: Red WIFI creada con Raspberry

Conexión de la Raspberry

Una vez creado el medio de comunicación con la Raspberry, podemos interactuar con ella de dos formas:

- Mediante terminal de comandos a través de SSH (Secure Shell).
- Por conexión, mediante escritorio remoto.

Este último método, es decir, por conexión mediante escritorio remoto, ha sido el empleado para el desarrollo del presente trabajo. La principal causa de nuestra elección ha sido que este método nos permite interactuar de forma gráfica con la Raspberry, sin necesidad de que esta esté conectada a una pantalla, a un teclado y a un ratón. En definitiva, el dispositivo SBC es controlado desde otro ordenador.

Cliente de escritorio

El cliente de escritorio remoto instalado en la Raspberry ha sido TightVNC. A continuación, se detalla brevemente la configuración realizada en él:

1. Instalamos el cliente con: `sudo apt-get install tightvncserver`

2. Ejecutamos el programa escribiendo en el terminal `TightVNC`.

3. Nos exige asignar una contraseña. Esta contraseña será la necesaria para conectarnos desde otro ordenador.

4. Iniciamos sesión en VNC en el escritorio 1 con resolución HD para monitores 1360x768

    - Comando VNC: `vncserver :1 -geometry 1360x768 -depth 24`

Configurado de esta manera, ya podremos conectarnos desde otro ordenador de manera remota utilizando un cliente de escritorio remoto. Cabe reseñar que, cada vez que iniciemos la Raspberry, se debe iniciar la sesión de escritorio remoto. Así pues, cada vez que la encendamos, tendremos que ejecutar el comando de inicio de sesión VNC anteriormente mencionado. Este proceso se ha automatizado para que se realice cada vez que se inicie la Raspberry, siguiendo las instrucciones que se muestran en la página oficial de la compañía[1].

Configuración de Remmina

Para realizar la conexión remota, se ha utilizado un ordenador portátil con el sistema operativo Ubuntu, en su versión 16.04 de 64 bits. La aplicación de cliente de escritorio ha sido Remmina, incluida en la configuración por defecto en esta versión de Ubuntu.

[1] Página fundación Raspberry pi, disponible en https://www.raspberrypi.org/

Figura 14: Logo escritorio remoto Remmina

La Figura 15 muestra la configuración necesaria de Remmina para la conexión con la Raspberry.

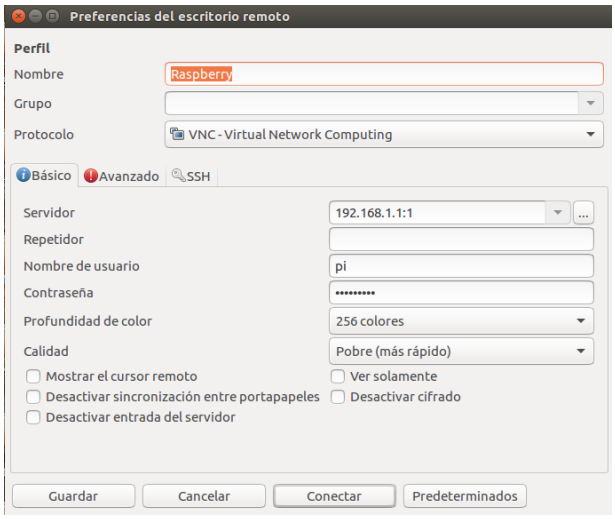

Figura 15: Configuración conexión por VNC a Raspberry mediante Remmina

En la Figura 15, se observa cómo se debe establecer un nombre para la conexión con la Raspberry. Este nombre no influye en la configuración de la conexión, únicamente sirve para distinguirla en el caso de que tengamos otras conexiones guardadas para otros dispositivos.

Los parámetros que sí influyen son el servidor, donde debemos introducir la dirección IP de la Raspberry seguida del separador (:), junto con el escritorio al que queremos conectarnos. En el paso anterior lo hemos configurado para que se mostrase el escritorio 1.

Además, debemos introducir el nombre de usuario, que por defecto en Raspbian es pi, junto con la contraseña establecida durante la configuración del cliente TightVNC en la Raspberry.

Con esos parámetros, ya podremos conectarnos a la Raspberry de manera remota desde un ordenador mediante escritorio remoto, utilizando la red WIFI que genera la propia Raspberry.

Módulo GNSS

El módulo GNSS utilizado es el RPI GPS ADD-ON V2.0 (Figura 16), y cuenta con las características especificadas en la Tabla 2.

Tabla 2: Especificaciones de RPI GPS ADD-ON V2.0

Tamaño	65mm X 56mm X 1.6mm
Voltaje Operativo	Digital 3.3V DC
Interface	UART
Velocidad de Transmision	9600(default)
Modulo GNSS	Ublox Neo 6 M

Figura 16: Modulo GNSS V2 para Raspberry

El módulo GNSS V2 para Raspberry utiliza el sensor Ublox Neo 6 M. Dispone de las especificaciones que se muestran en la Figura 17:

Parameter	Specification			
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS			
Time-To-First-Fix[1]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Cold Start[2]	26 s	27 s	32 s
	Warm Start[2]	26 s	27 s	32 s
	Hot Start[2]	1 s	1 s	1 s
	Aided Starts[3]	1 s	<3 s	<3 s
Sensitivity[4]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Tracking & Navigation	-162 dBm	-161 dBm	-160 dBm
	Reacquisition[5]	-160 dBm	-160 dBm	-160 dBm
	Cold Start (without aiding)	-148 dBm	-147 dBm	-146 dBm
	Hot Start	-157 dBm	-156 dBm	-155 dBm
Maximum Navigation update rate		NEO-6G/Q/M/T	NEO-6P/V	
		5Hz	1 Hz	
Horizontal position accuracy[6]	GPS	2.5 m		
	SBAS	2.0 m		
	SBAS + PPP[7]	< 1 m (2D, R50)[8]		
	SBAS + PPP[7]	< 2 m (3D, R50)[8]		
Configurable Timepulse frequency range		NEO-6G/Q/M/P/V	NEO-6T	
		0.25 Hz to 1 kHz	0.25 Hz to 10 MHz	
Accuracy for Timepulse signal	RMS	30 ns		
	99%	<60 ns		
	Granularity	21 ns		
	Compensated[9]	15 ns		
Velocity accuracy[6]		0.1m/s		
Heading accuracy[6]		0.5 degrees		
Operational Limits	Dynamics	≤ 4 g		
	Altitude[10]	50,000 m		
	Velocity[10]	500 m/s		

[1] All satellites at -130 dBm
[2] Without aiding
[3] Dependent on aiding data connection speed and latency
[4] Demonstrated with a good active antenna
[5] For an outage duration ≤10s
[6] CEP, 50%, 24 hours static, -130dBm, SEP: <3.5m
[7] NEO-6P only
[8] Demonstrated under following conditions: 24 hours, stationary, first 600 seconds of data discarded. HDOP < 1.5 during measurement period, strong signals. Continuous availability of valid SBAS correction data during full test period.
[9] Quantization error information can be used with NEO-6T to compensate the granularity related error of the timepulse signal
[10] Assuming Airborne <4g platform

Figura 17: Especificaciones GNSS Ublox familia NEO-6[2].

El modelo empleado utiliza un receptor GPS con solución de código. Además, es compatible con sistemas de aumentación o SBAS, lo que nos proporciona una precisión en posicionamiento horizontal en torno a unos 2 metros en condiciones óptimas, es decir, con un horizonte despejado de obstáculos y en condiciones meteorológicas favorables.

[2] Fuente: www.u-blox.com

Además, toda la familia NEO-6, incluye GPS asistido (A-GPS) que, como se comentó en el apartado de los sistemas de navegación, proporciona información de los satélites efemérides y del almanaque, reduciendo así los tiempos de inicialización para obtener la posición.

Configuración del módulo en la Raspberry

A continuación, se detalla el proceso de configuración llevado a cabo para el funcionamiento del módulo en la Raspberry, para la versión de Rapbian Jessie:

En primer lugar, se debe activar el puerto serie que esta desactivado por defecto. Para ello, abrimos las opciones avanzadas del menú de configuración de la Raspberry en Raspbian, y lo activamos, tal y como se muestra en la Figura 18:

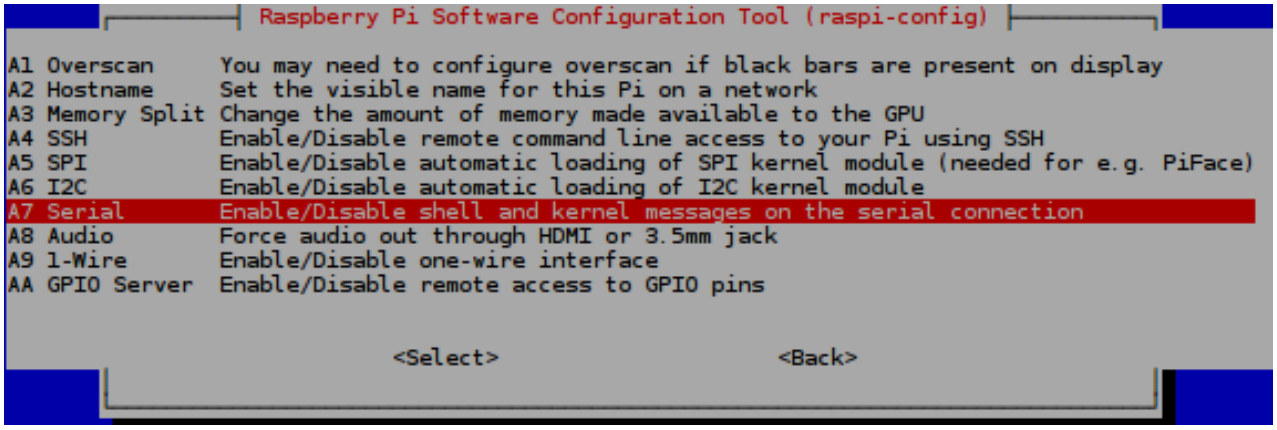

Figura 18: Opciones avanzadas del menú de configuración Raspbian.

Una vez activado el puerto serie necesario para el funcionamiento del módulo GNSS, debemos instalar los clientes GPSD y la librería de Python para GPS escribiendo en el terminal:

```
sudo apt-get install gpsd gpsd-clients python-gps
```

GPSD es lo que en jerga UNIX se denomina daemon o demonio , es decir, una aplicación o programa que se ejecuta en segundo plano y que recoge los datos del módulo GNSS y los envía a través de un socket, es decir, a través de un mecanismo que permite la conexión entre distintos procesos.

Previamente, tendremos que modificar el archivo alojado en */etc/default/gpsd* con el editor de textos nano escribiendo en el terminal:

```
sudo nano /etc/default/gpsd
```

Quedando de la siguiente forma:

```
DEVICES=”/dev/ttyAMA0”
```

Tras esto, debemos activar el cliente GPSD mediante terminal con:

```
sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock
```

Esta acción se debe realizar cada vez que se inicia la Raspberry, al igual que ocurría con el cliente de escritorio remoto. Este proceso también se ha automatizado siguiendo las instrucciones que se pueden encontrar en diversos sitios web [3]

Una vez realizado el anterior proceso, podemos comprobar nuestra posición por GNSS escribiendo *gpsmon* en el terminal o a través de scripts de Python, utilizando las librerías de Python para GPS instaladas anteriormente (Figura 19).

[3] Francesc. (2014). Tracker GPS con raspberry pi y el módulo ublox M6, disponible en http://fpaez.com/tracker-gps-con-raspberry-pi/

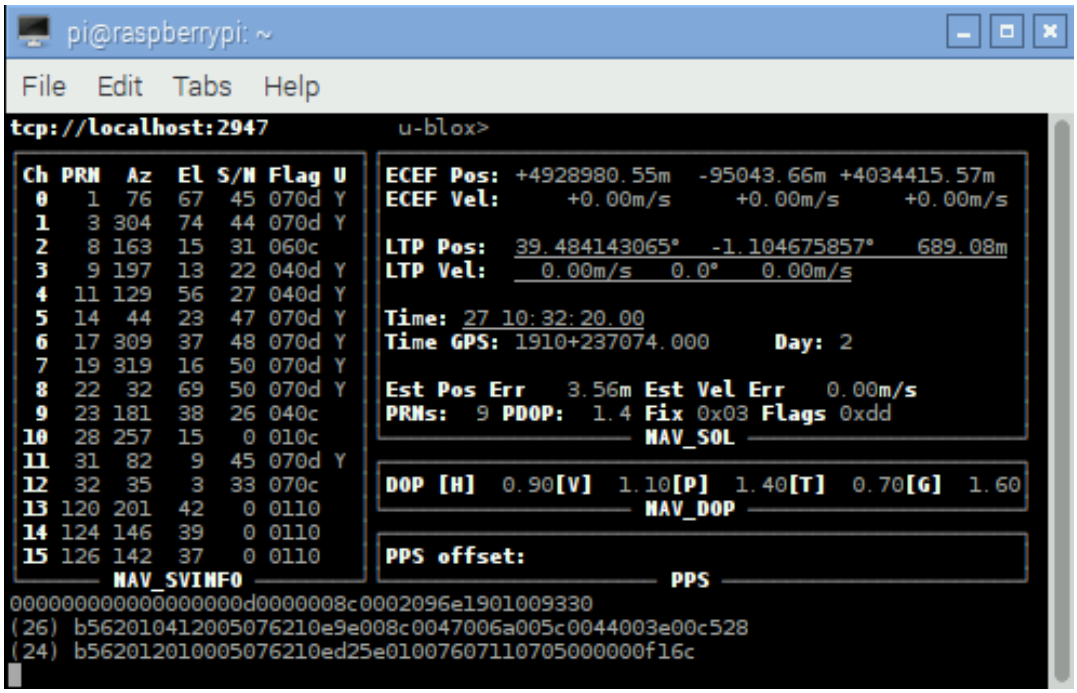

Figura 19: Interfaz del programa GPSMON en terminal en Raspberry.

5. VEHÍCULOS

Para la realización del presente trabajo se han diseñado y construido dos vehículos, denominados ROVER 1 y ROVER 2, cada uno de ellos con diferentes características. En este apartado se presentan ambos, las piezas que los componen y sus características.

El “cerebro” de ambos vehículos es el mismo, la Raspberry Pi 2 junto con el módulo GNSS y el receptor WIFI .Todos estos elementos están alimentados mediante una batería externa o powerbank de 8000 mAh de capacidad, lo que nos asegura varias horas de uso ininterrumpido sin necesidad de recargarla.

Figura 20: Powerbank 8000mah marca Celly.

La powerbank dispone de dos salidas, una con un voltaje de salida de 5 Voltios y 1 Amperio y otra con 5 Voltios y 2.1 Amperios. Como la Raspberry tiene conectados el módulo GNSS y el receptor WIFI, el consumo eléctrico aumenta, pudiendo ser escasa la potencia que ofrece la salida de 1 amperio. Así pues, es recomendable utilizar la salida que ofrece 2.1 amperios para que, en momentos de alta demanda energética, no se produzcan reinicios de la Raspberry por falta de energía.

ROVER 1

Las Figuras 21, 22 y 23, muestran el primer vehículo construido y sus componentes:

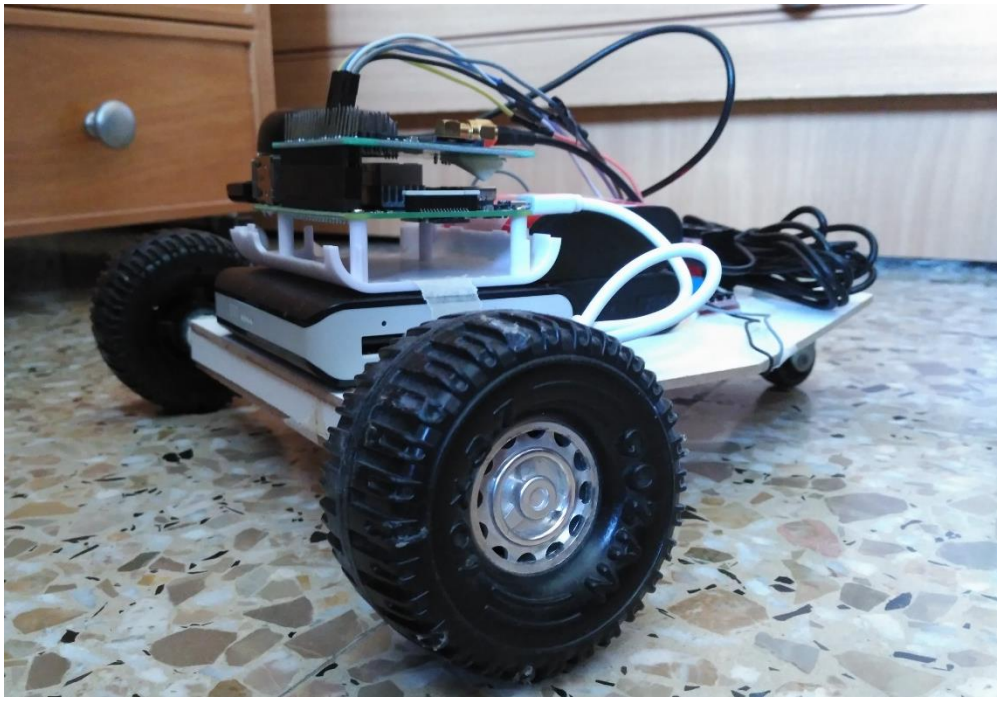

Figura 21: ROVER 1.

El primer vehículo, o ROVER 1, dispone de una configuración de vehículo o robot cinemático diferencial con dos ruedas en el mismo eje, con un motor para cada rueda, y en la parte posterior una rueda loca.

Figura 22: ROVER 1, vista de perfil.

Figura 23: ROVER 1, vista trasera.

El ROVER 1 está configurado de la siguiente forma:

- ✓ Dos ruedas motrices en la parte delantera, unidas cada una de ellas a unos pequeños motores.
- ✓ Los motores, a su vez, están unidos al chasis o cuerpo del ROVER, el cual se constituye de una pieza de madera de aglomerado.
- ✓ En la parte posterior se halla la rueda loca. Su única función es la de hacer girar la parte trasera del vehículo.

DRIVER

Para controlar los motores se ha empleado el driver L298N, el cual permite controlar los dos motores la vez.

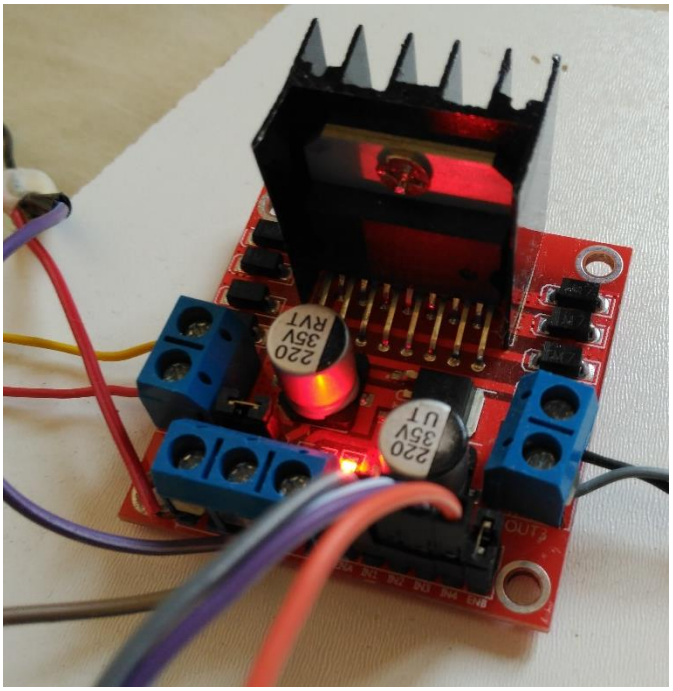

Figura 24: Controlador de motores dual L298N.

El driver posee las siguientes características:

- ➢ Voltaje: Entre 6 y 35 V en corriente continua.
- ➢ Corriente máxima: 2 Amperios.

- Posee un regulador LM7805 integrado para alimentar la parte lógica, con lo que se puede alimentar la tarjeta con 12V, por ejemplo.
- Posee 6 entradas de control (ver tabla de control).
- Admite entradas de señal PWM para el control de velocidad.
- Tamaño: 43 mm x 23,9 mm x 43 mm.
- Salidas: para 2 motores de DC o para un motor bipolar paso a paso.

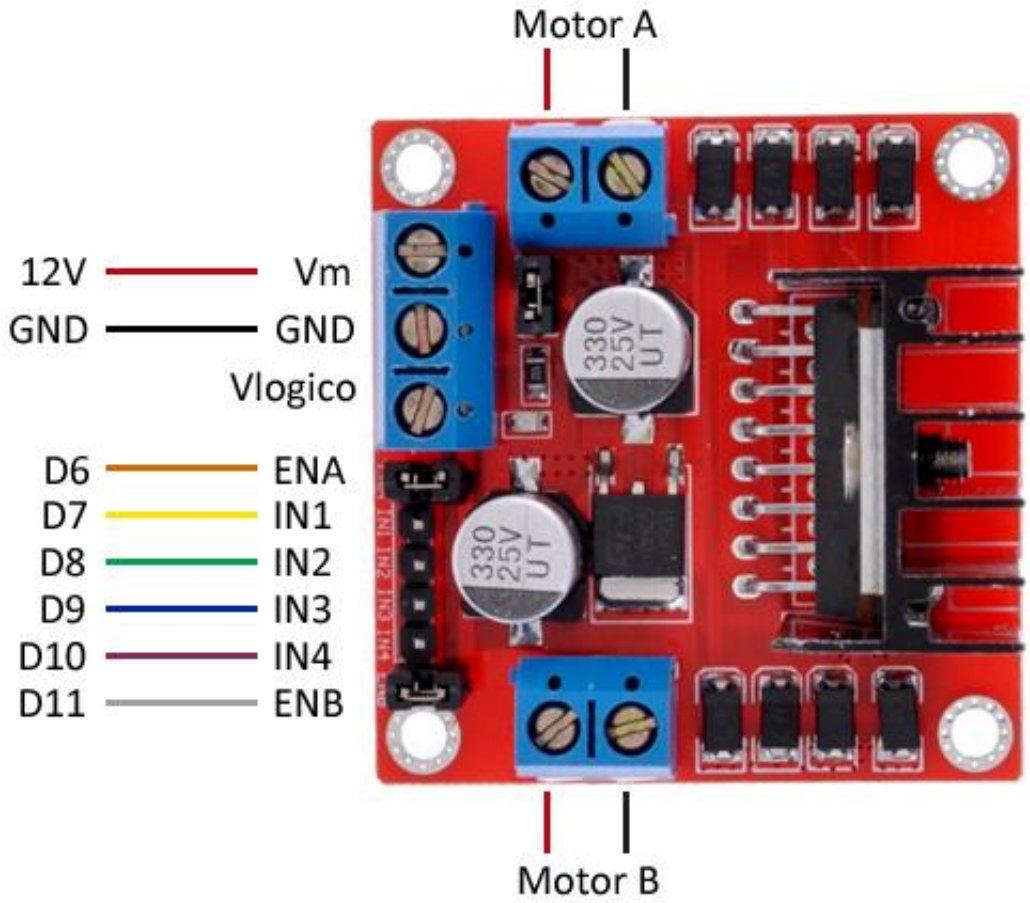

Figura 25: Esquema conexiones controlador motores L298N. Fuente: www.luisllamas.es.

ALIMENTACIÓN DEL DRIVER Y LOS MOTORES

Para alimentar el driver y los motores, se han utilizado dos pilas de petaca conectadas en serie, consiguiendo así un voltaje de 9V. Las pilas han sido conectadas a la entrada de 12V del driver.

PINES

Los pines de entrada del controlador se han conectado a los puertos GPIO de la Raspberry.

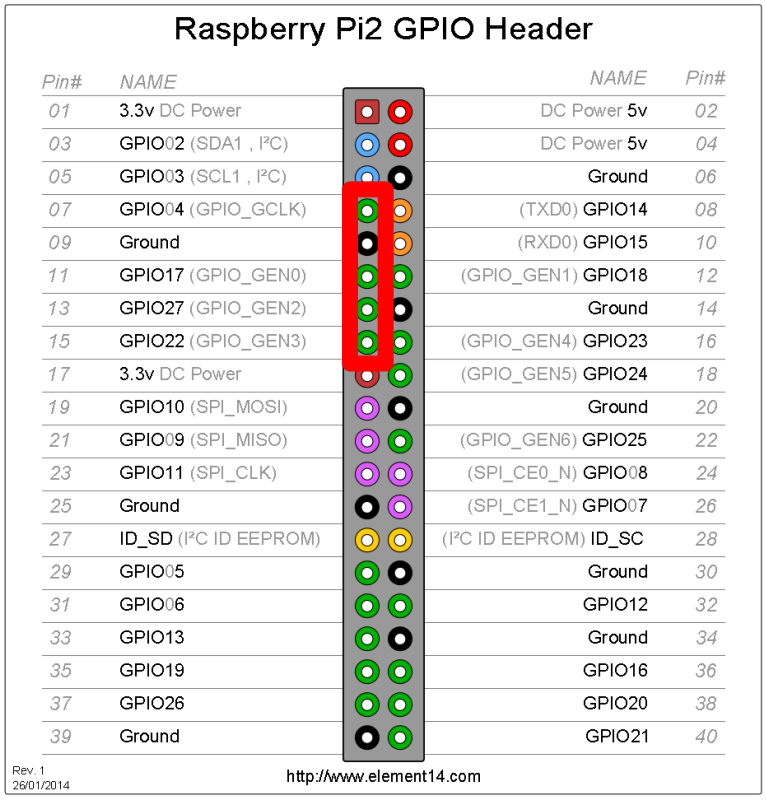

Figura 26: Pines empleados para conectar el controlador de motores.

Se han empleado los pines GPIO04, GPIO17, GPIO27, GPIO22 conectados a IN1, IN2, IN3 e IN4 respectivamente. Las entradas 1 y 2 controlan el motor derecho y las entradas 3 y 4 controlan el motor izquierdo.

MOTORES UTILIZADOS

Figura 27: Motores del ROVER 1.

Conectados al controlador, se encuentran los dos pequeños motores con caja reductora que cuentan con las siguientes características:

- Tensión nominal: DC 6V, corriente nominal: 0.3A (sin carga).
- Potencia: Sin carga 1.8W.
- Velocidad: 60 RPM sin carga.
- Diámetro Del Eje: 3mm/ 0.3cm.
- Tamaño caja de cambios: 12 x 10 x 16mm/ Tamaño: 12 x 10 x 36mm
- Peso: 10g.

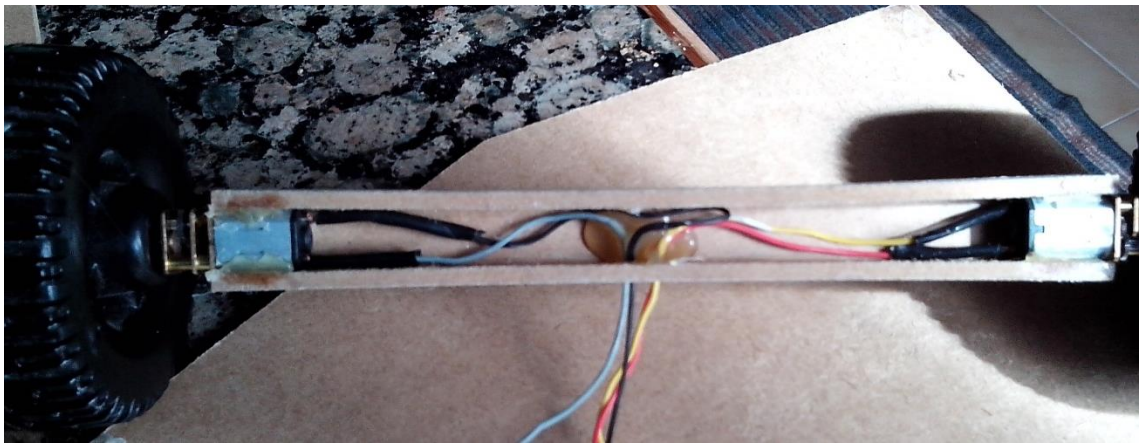

Figura 28: Instalación de los motores del eje del ROVER 1.

Así pues, ya tendríamos todos los elementos y conocimientos para el diseño y construcción del vehículo ROVER 1. Cabe reseñar el bajo coste de los materiales del vehículo, ya que el controlador de los motores tiene un precio inferior a 2€ y ambos motores se encuentran fácilmente en Internet por unos 12€.

Presentado el primer vehículo, realizaremos el procedimiento análogo con el segundo vehículo construido.

ROVER 2

En las Figuras 29 y 30, se muestra el segundo vehículo construido y sus componentes:

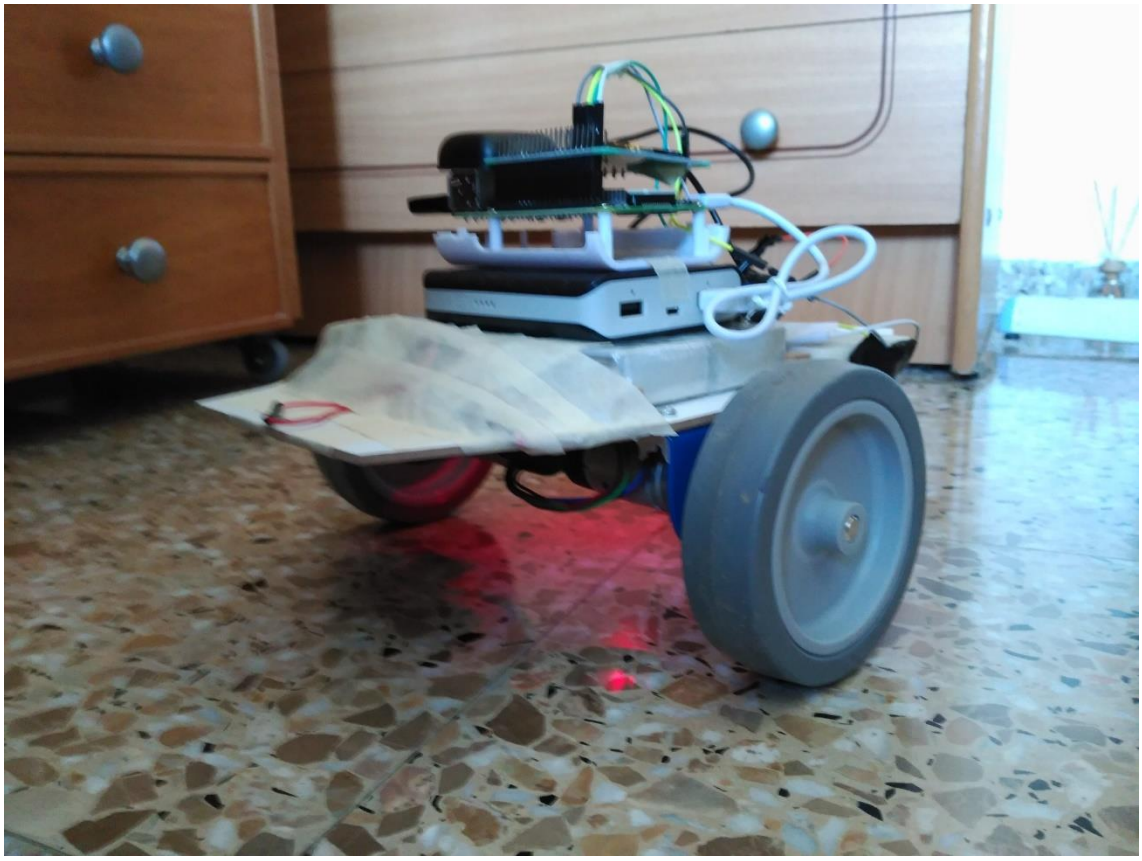

Figura 29: ROVER 2.

El segundo vehículo, o ROVER 2 dispone, al igual que su antecesor, de una configuración de vehículo o robot cinemático diferencial con dos ruedas en el mismo eje, con un motor para cada rueda, y en la parte posterior una rueda loca.

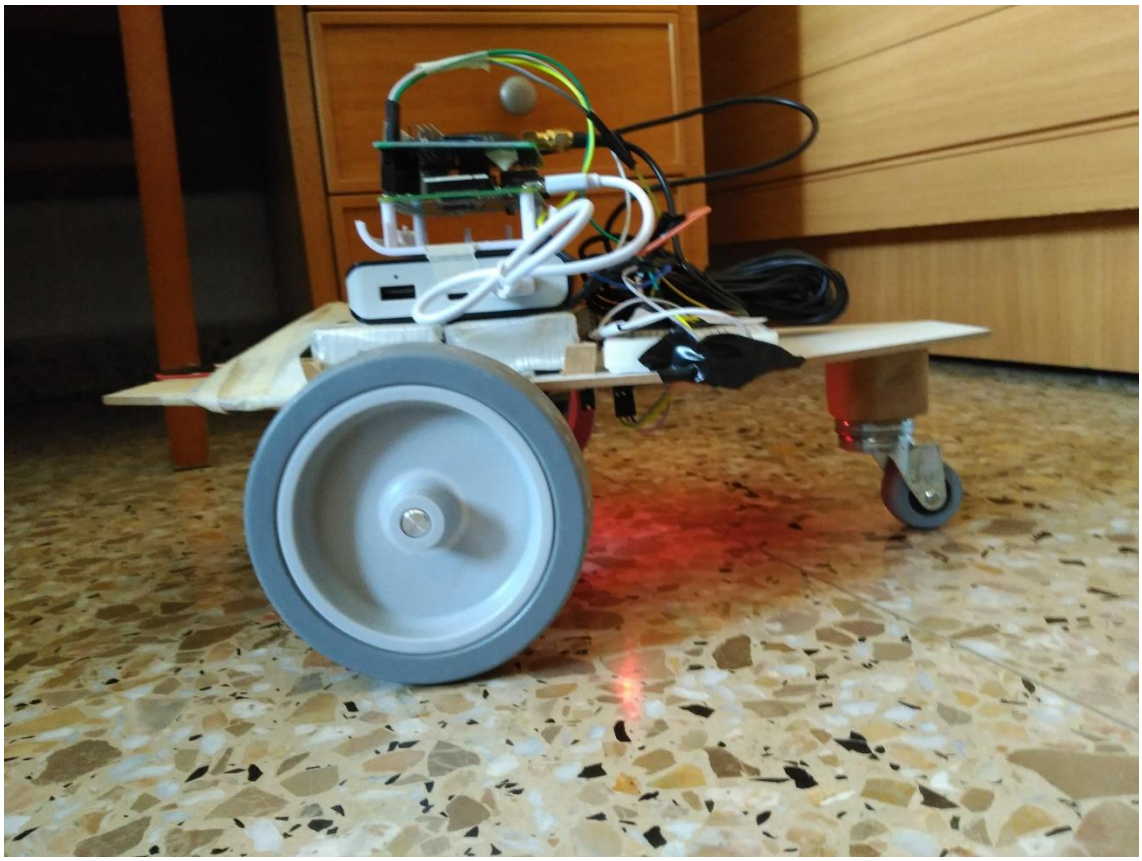

Figura 30: ROVER 2, vista lateral.

A continuación, se detallan los materiales utilizados en el vehículo:

KIT

El ROVER 2 utiliza un kit de locomoción llamado RD02[4]. El RD02 es un kit de locomoción completo para robots, el cual incluye unos motores con encoder que proporcionan una gran precisión sobre el movimiento real de las ruedas. Además, el kit incluye todo lo necesario para mover un robot de tamaño pequeño o mediano de una forma muy sencilla. Cabe reseñar que es necesario disponer de una alimentación única de 12 V, tanto para los motores como para el circuito de control.

[4] http://www.superrobotica.com.

El kit está compuesto por (Figura 31):

- El circuito de control MD25.
- Dos motores con encoder EMG30 de 12 V.
- Dos soportes de motor
- Dos ruedas de 100 mm de diámetro.

El circuito se controla tanto mediante línea serie o por bus I2C, e incluye registros específicos para asignar fácilmente dirección, velocidad, aceleración, e incluso para leer el consumo de corriente de los motores.

Figura 31: Kit de locomoción RD02. Fuente: www.superrobotica.com

BATERÍAS

El ROVER 2 incluye dos baterías de polímeros de litio con una tensión de salida de 12 voltios y una capacidad de 2600 mAh, conectadas en paralelo, consiguiendo así una capacidad conjunta de 5200mah y un voltaje de salida de 12 voltios (Figura 32).

Figura 32: Kit locomoción y baterías ROVER 2.

CONTROLADOR DE MOTORES MD25

A continuación, se muestran las características y especificaciones del controlador de motores MD25, extraídas de la hoja de especificaciones:

Se trata de un potente circuito controlador dual de motor a través de bus serie e I2C, diseñado para su uso con el motor EMG30 (Motor DC reductor 12 voltios 170 RPM con codificador EMG30). Las características más destacadas del producto son:

- ✓ Lee los codificadores de los motores y proporciona recuentos en los registros para determinar la distancia cubierta y la dirección.

- ✓ Puede controlar dos motores de forma independiente o conjunta.
- ✓ La corriente del motor puede leerse a través de los registros.

- ✓ La alimentación del módulo es única, de 12V.

- ✓ El regulador de 5V en placa puede proporcionar hasta 1A de pico o 300 mA de forma continua para los circuitos externos.

- ✓ Su función de control de dirección puede controlarse mediante un simple comando.

- ✓ Permite la regulación de la aceleración, así como de la potencia.

TENSIÓN DE MOTOR

El MD25 está diseñado para trabajar en un rango de tensión entre los 9 y los 14V. Una tensión por debajo de los 9V garantiza, prácticamente con total seguridad, que los motores no se moverán.

SUPRESIÓN DEL RUIDO DEL MOTOR

Al utilizar los motores EMG30 con codificador, se podrá comprobar que se ha añadido un condensador de supresión de ruido de 10 nF. El condensador debería ser capaz de manejar una tensión dos veces superior a la tensión del motor.

INDICADORES LUMINOSOS

El vehículo dispone de dos indicadores luminosos, uno de color rojo y otro verde, los cuales tienen distinta finalidad:

- ➢ El led de alimentación de color rojo indica que se ha aplicado la alimentación al módulo.

- El led de color verde parpadea para indicar, mediante destellos, la dirección I2C. Después, se ilumina durante 500 ms para indicar que la actividad del bus I2C es correcta en el módulo. Además, el temporizador interno de 500 ms se pone a cero cada vez que hay actividad en el bus I2C, por lo que estará activo durante un acceso continuo.

CONEXIONES

En la Figura 33 se pueden identificar las diferentes conexiones del circuito MD25:

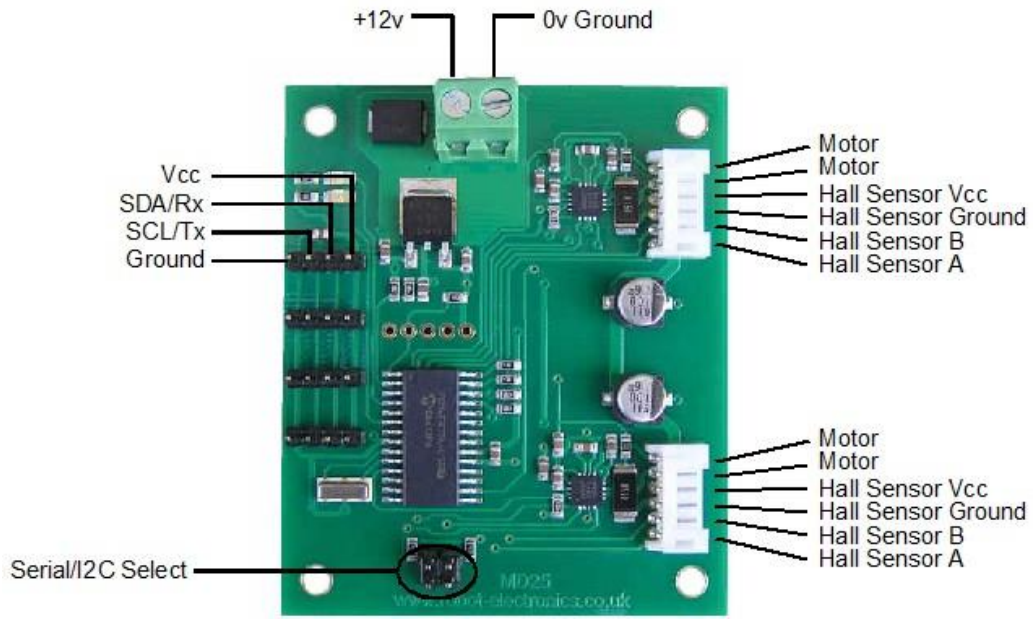

Figura 33: Esquema Controlador MD25. Fuente: www.superrobotica.com

REGULACIÓN AUTOMÁTICA DE LA VELOCIDAD

El controlador MD25 puede aumentar de forma dinámica la potencia, recibiendo la información de los codificadores según sea necesario. Si no se lograse la velocidad requerida, el controlador MD25 aumentará la potencia de los motores, hasta alcanzar la velocidad deseada o hasta que se alcance la

salida máxima. La regulación de velocidad puede desactivarse a través del registro de comandos.

PARADA AUTOMÁTICA DEL MOTOR

El controlador MD25 detendrá automáticamente los motores en el caso de que no existan comunicaciones I2C en 2 segundos. Esta función sirve para evitar que el robot pierda el control en caso de que el controlador falle. Esta función puede desactivarse, en caso necesario.

CONTROL DEL MD25

El controlador MD25 está diseñado para operar en un sistema de bus I2C estándar en las direcciones entre 0xB0 y 0xBE (el último bit de la dirección es un bit de lectura/escritura, por lo que sólo pueden ser números pares), siendo 0xB0 la dirección por defecto. Se puede modificar con facilidad por software.

El modo I2C permite al controlador MD25 conectarse a la mayoría de los controladores de mercado tales como PICAXE, BasicX24 y BS2P, y una amplia gama de microcontroladores como PIC, AVR, 8051, etc.

El protocolo de comunicación I2C con el módulo MD25 es el 24C04 Que opera como sigue. Para leer uno o más registros de MD25, primero se envía el bit de inicio, la dirección del módulo (0XB0 por ejemplo) con el bit de nivel lógico bajo de lectura/escritura, y después el número de registros que desea leer. Esto va seguido de un inicio repetido, y de nuevo por la dirección del módulo con el bit lógico alto de lectura/escritura (0XB1 en este ejemplo). Ahora ya puede leer uno o más registros. El controlador MD25 tiene 17 registros numerados del 0 al 16 de la manera que se detalla en la Tabla 3:

Tabla 3: Registros Controlador MD25. Fuente: www.superrobotica.com

Registro	Nombre	Lectura/Escritura	Descripción
0	Velocidad1	R/W (Lectura/Escritura)	Velocidad de Motor1 (modo 0,1) o velocidad (modo 2,3)
1	Velocidad2/Giro	R/W (Lectura/Escritura)	Velocidad de Motor2 (modo 0,1) o velocidad (modo 2,3)
2	Codificador1a	Sólo lectura	La posición del Codificador 1, el 1º byte (nivel lógico más alto) y recuento de la captura durante la lectura
3	Codificador1b	Sólo lectura	Posición del Codificador 1, 2º byte
4	Codificador1c	Sólo lectura	Posición del Codificador 1, 3º byte
5	Codificador1d	Sólo lectura	Posición del Codificador 1, 3º byte
6	Codificador2a	Sólo lectura	La posición del Codificador 2, el 1º byte (nivel lógico más alto) y recuento de la captura durante la lectura
7	Codificador2b	Sólo lectura	Posición del Codificador 2, 2º byte
8	Codificador2c	Sólo lectura	Posición del Codificador 2, 3º byte
9	Codificador2d	Sólo lectura	Posición 2 del Codificador 4º byte (byte de nivel lógico más bajo)
10	Voltios de la batería	Sólo lectura	Tensión de alimentación
11	Corriente del Motor 1	Sólo lectura	La corriente del motor 1
12	Corriente del Motor 2	Sólo lectura	La corriente del motor 2
13	Revisión de software	Sólo lectura	Número de revisión de software
14	Tasa de aceleración	R/W (Lectura/Escritura)	Registro opcional de aceleración
15	Modo	R/W (Lectura/Escritura)	Modo de operación (vea a continuación)
16	Comando	R/W (Lectura/Escritura)	Utilizado para poner a cero los recuentos del codificador y cambiar la dirección del módulo

REGISTRO DE MODO

El controlador puede funcionar en varios modos, los cuales se detallan a continuación.

El registro de modo selecciona el modo de operación y el tipo de entrada de datos I2C que requiere el usuario. Las opciones son:

- 0: (Valor por defecto) Si se escribe un valor 0 en el registro de modo, entonces el significado de la velocidad será una velocidad literal en el rango de 0 (Marcha atrás) 128 (Parada) 255 (Marcha hacia delante).

- 1: El Modo 1 es similar al Modo 0, salvo en que los registros de velocidad se interpretan como valores con signo. El significado de los registros de velocidad es la velocidad literal en el rango -128 (Marcha hacia delante), 0 (Parada) y 127 (Marcha hacia delante).

- 2: Si se escribe un valor 2 en el registro de modo, la velocidad 1 controlará la velocidad de ambos motores, y velocidad 2 se convierte en la velocidad de giro. Los datos están en el rango entre 0 (Marcha hacia atrás), 128 (Parada) y 255 (Marcha hacia delante).

- 3: El Modo 3 es similar al Modo 2, salvo en que los registros de velocidad se interpretan como valores con signo. Los datos están en el rango entre -128 (Marcha hacia atrás), 0 (Parada) y 127 (Marcha hacia adelante)

COMPORTAMIENTO DEL CONTROLADOR

En la Tabla 4, se muestran los comandos para el manejo del comportamiento del controlador.

Tabla 4: Comandos del controlador MD25. Fuente: www.superrobotica.com

Comando		Acción
Dec	Hex	
32	20	Pone a cero los registros del codificador
48	30	Desactiva la regulación automática de velocidad
49	31	Activa la regulación automática de velocidad (por defecto)
50	32	Desactiva el tiempo de inactividad de 2 segundos de los motores (a partir de la Versión 2)

51	33	Activa el tiempo de inactividad de 2 segundos de los motores (por defecto) cuando no hay comunicaciones por el bus I2c (a partir de la Versión 2)
160	A0	1º en la secuencia para modificar la dirección I2C
170	AA	2º en la secuencia para modificar la dirección I2C
165	A5	3º en la secuencia para modificar la dirección I2C

Estos comandos y registros se utilizan para controlar los motores utilizando los pines I2C de la Raspberry, mediante la librería de Python smbus. Un I2C o circuito interintegrado, es un bus serie de datos que se utiliza principalmente a nivel interno para la comunicación entre diferentes partes de un circuito. El I2C utiliza dos líneas, una para transmitir los datos (SDA), y otra para el reloj asíncrono, que indica cuando leer los datos llamada (SCL).

RESISTENCIAS

Las líneas I2C de la Raspberry necesitan estar conectadas a resistencias de 10 kΩ, en configuración pull-up hacia la salida de corriente de 3.3 Voltios de la Raspberry (Figura 34).

La función principal de la resistencia pull-up es evitar que una corriente excesiva fluya a través del circuito. Además, esta resistencia también es la encargada de evitar que haya ruido en la señal, el cual provocaría lecturas erróneas.

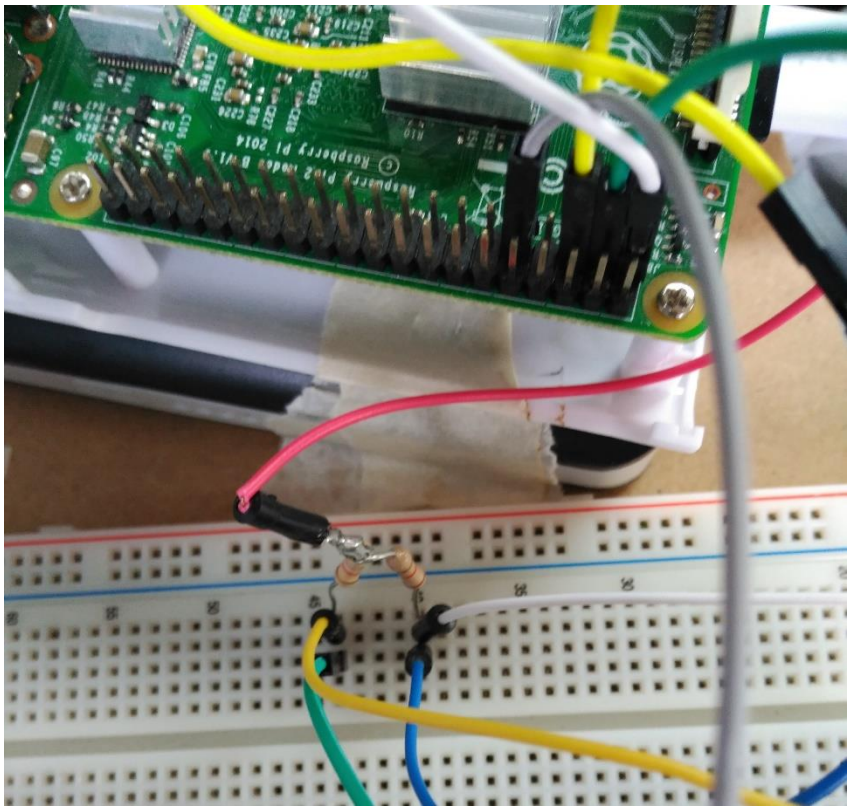

Figura 34: Configuración pull-up SDA y SCL de la Rapberry.

ROVER 1 VS ROVER 2

La principal diferencia del ROVER 2 respecto al ROVER 1 es que el segundo prototipo, es decir, el ROVER2, hace uso de encoders o codificadores en sus motores. Estos encoders son los encargados de medir las vueltas que da cada motor, posibilitándonos de esta forma, calcular la distancia recorrida y la velocidad. Esto nos permite hacer uso de la estimación de la posición mediante odometría, la cual combinaremos con la navegación por satélite.

Además el driver MD25, utilizado en el ROVER 2 pero no en el ROVER 1, proporciona la opción de controlar la velocidad de forma automática, elemento imprescindible en las ecuaciones de cinemática diferencial empleadas.

Por tanto, estas mejoras dotan al ROVER 2 de unas mayores aptitudes para la navegación autónoma frente al ROVER 1, el cual solo dispone de posicionamiento por satélite para su navegación.

Cabe reseñar que estas mejoras conllevan un incremento en el precio de sus elementos y por tanto, evidentemente, un incremento en el coste global del vehículo.

Explicadas las características y componentes del ROVER 2, y construido éste, nos encontramos en disposición de elaborar los programas en Python para posibilitar la navegación de ambos vehículos.

6. SOFTWARE DESARROLLADO

En este apartado se describirán los diferentes programas desarrollados en Python para los dos vehículos:

Scripts ROVER 1

El ROVER 1 fue el primer vehículo diseñado. Como se comentó anteriormente, este no dispone de codificadores en los motores, lo que lo incapacita para estimar su posición por odometría. Por tanto, solo se puede desplazar de forma forma autónoma mediante la información procedente del GNSS.

Para el ROVER 1 se han desarrollado dos scripts o programas llamados ROVER1_control_manual.py y ROVER1_control_automatico.py.

El script ROVER1_control_manual.py, nos permite controlar el vehículo de forma manual de manera remota. Desde el ordenador, que está conectado a la Raspberry, podemos controlar el ROVER, haciendo uso de las teclas a,w,s,d,q,e del teclado, tal y como se muestra en la Figura 35:

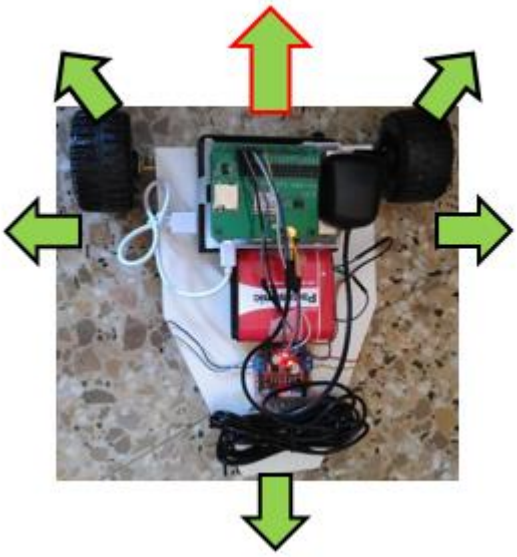

Figura 35: Movimientos del ROVER 1

Pulsando la tecla (w) avanza hacia adelante, y con la (d) y (a) gira hacia la derecha e izquierda respectivamente sobre sí mismo.

Pulsando (e) avanza hacia adelante pero girando hacia la derecha, y hacia la izquierda si pulsamos (q). Finalmente, si queremos retroceder pulsaremos la tecla (s).

El script ROVER1_control_automatico.py pretende que el ROVER 1 recorra un itinerario de manera autónoma, únicamente empleando el sensor GNSS acoplado a la Raspberry y conociendo previamente las coordenadas del itinerario a recorrer.

La idea de funcionamiento es sencilla, buscar siempre el punto más próximo a nuestra posición e ir hacia él.

Para saber si el vehículo debe girar hacia la derecha o izquierda, se hace uso de tres puntos singulares, los cuales muestran en la Figura 36:

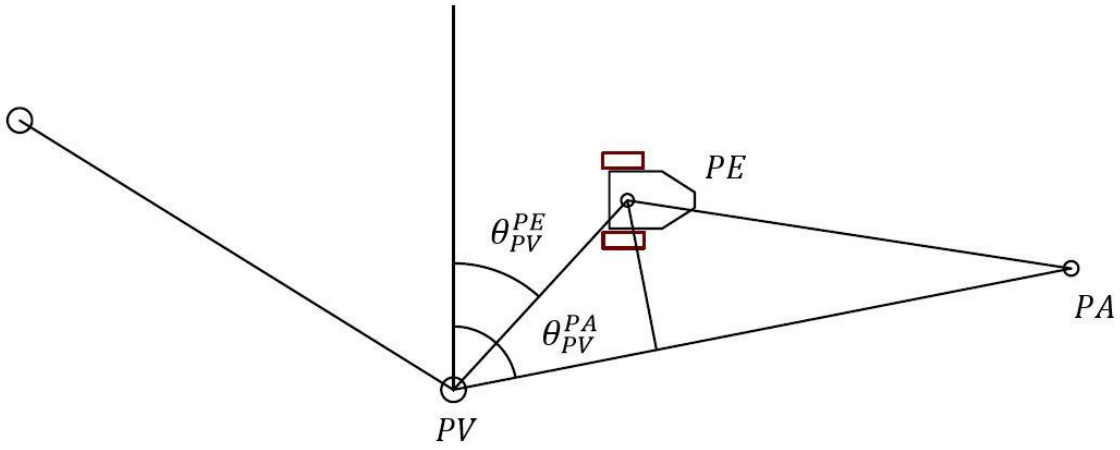

Figura 36: Esquema navegación ROVER 1.

El punto hacia el que vamos es el punto visado (PV), el punto en el que estamos es el punto estación (PE) en cada instante y (PA) es un punto auxiliar, que puede ser el punto inicial desde el que se parte el vehículo, o cada vez que llegados a un vértice de itinerario, el punto del vértice en el que nos encontramos.

Con estos tres puntos se calcula el azimut entre el vehículo y el punto objetivo θ_{PV}^{PE}, y el azimut entre el punto de auxiliar y el punto visado θ_{PV}^{PA} . A la diferencia entre estos dos ángulos se le ha llamado convergencia.

$$Convergencia = \theta_{PV}^{PA} - \theta_{PV}^{PE}$$

Si la convergencia es positiva, el vehículo girará hacia la izquierda y si es negativa, girará hacia la derecha.

Además del ángulo, se utiliza el criterio de minimizar la distancia perpendicular al eje PA-PV, partiendo de la idea de que la distancia perpendicular al eje debe ser menor a la proyección de la distancia hasta el punto en el eje PA-PV.

Cabe reseñar que las pruebas realizadas con con el ROVER 1, demostraron que esta idea de navegación autónoma solo con sistemas de navegación por satélite, obliga a que la precisión en posicionamiento sea muy alta, cosa que el modulo acoplado a la Raspberry no ofrece.

Si además de esto unimos la baja velocidad que tiene el ROVER 1, la navegación autónoma con este vehículo se vuelve una tarea muy difícil por no decir casi imposible, razón por la cual se decidió utilizar más de una técnica para el posicionamiento y diseñar el ROVER 2.

Script ROVER 2

Para el ROVER 2 se ha creado un script que emplea posicionamiento combinando por GNSS y odometría, y calcula la trayectoria entre su posición y el punto objetivo utilizando el algoritmo de persecución.

En la Figura 37, se muestra el flujo de trabajo del script.

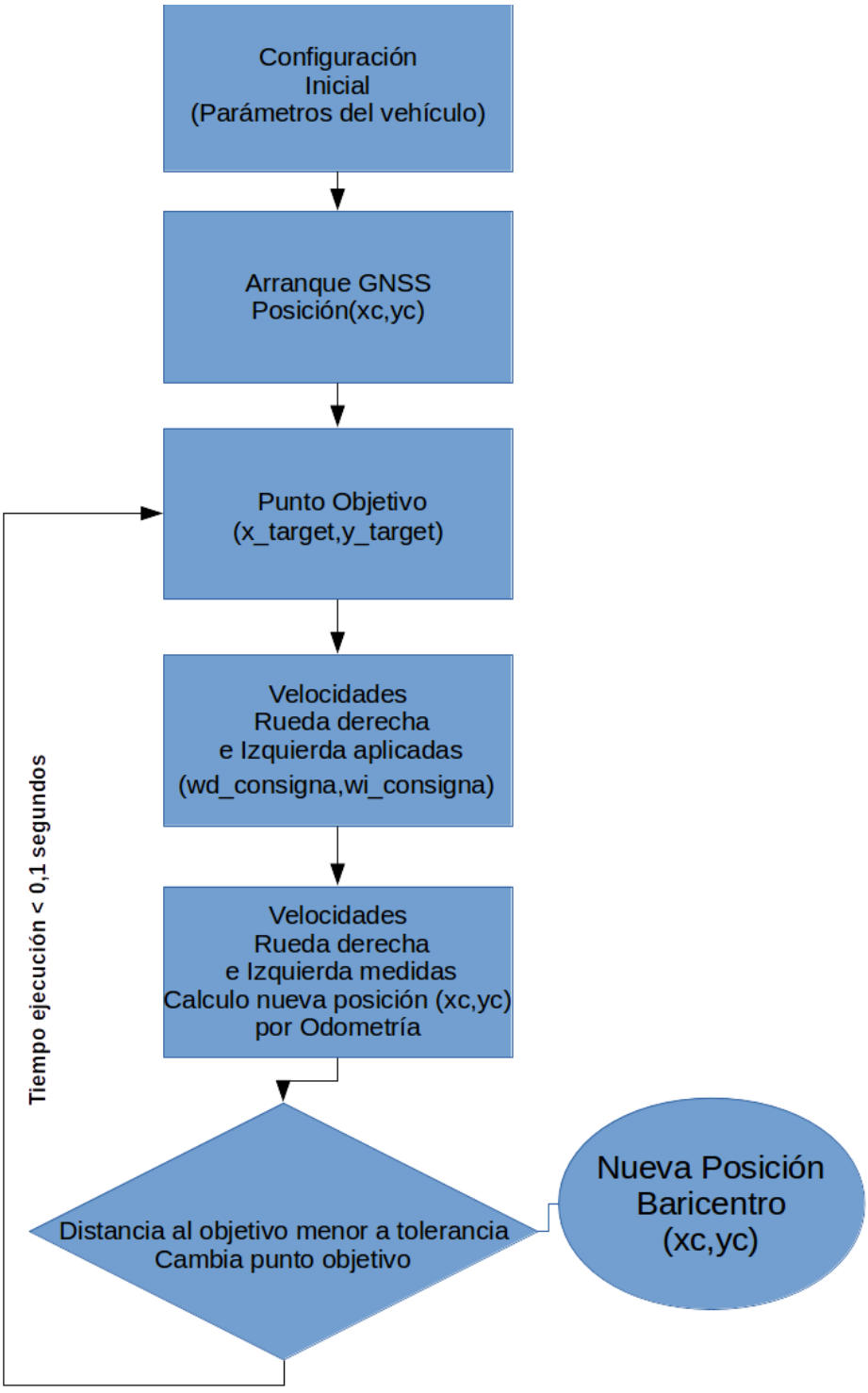

Figura 37: Esquema funcionamiento ROVER 2.

A continuación, se detallan los pasos a del funcionamiento del ROVER 2:

- El primer paso es inicializar las variables para configurar el driver MD25 mediante la librería smbus, con la cual podemos comunicarnos mediante i2c desde la Raspberry al controlador de motores. Además, activamos el control de velocidad del driver, para que la velocidad de avance del vehículo sea constante.

- Se define la ruta o itinerario a seguir por el vehículo. Este itinerario está almacenado en una lista que contiene para cada registro el número del punto junto con sus coordenadas x e y en la proyección UTM en el sistema de referencia ETRS89.

- Se establece una distancia mínima para el cambio de punto objetivo, es decir, cuando el vehículo este por debajo de esa distancia, cambiará el punto al que debe ir por el siguiente en la ruta.

- Se añaden los parámetros del vehículo tales como la distancia entre las ruedas y radio efectivo de las ruedas.

- Se establece la velocidad de avance del vehículo en metros por segundo.

- Se establece el ángulo que forma el vehículo con respecto al eje de las x. Para hacer coincidente el sistema de coordenadas del vehículo y el de los puntos de la ruta el vehículo se orienta con respecto al Este, que coincide con el eje de las x en el sistema de proyección UTM.

- Una vez inicializadas las variables, se inicia el sensor GNSS haciendo uso de la librería para GPSD para Python. Para mejorar la navegación, se inicia el GPS y se espera un periodo de tiempo desde el cálculo de la primera posición GPS o Time To First Fix (TTFF), con objeto de mejorar la precisión en posicionamiento.

- Transcurrido este periodo de tiempo, se capturan las coordenadas GNSS de ese punto y se transforman a la proyección UTM mediante la librería pyproj. Esas coordenadas serán las del punto de inicio del vehículo.

A continuación se da comienzo a la iteración para recorrer la ruta. En ella se realizan las siguientes tareas:

- Se calculan las velocidades angulares de las ruedas derecha e izquierda, aplicando las formulas del algoritmo de persecución pura.

- Estas velocidades angulares son enviadas al controlador de motores en forma de byte.

- A continuación, se leen las velocidades reales registradas por los codificadores de los motores y se calcula la nueva posición del vehículo por medio de las fórmulas de odometría.

- Si la nueva posición está a una distancia menor a la establecida previamente, se pasa al siguiente punto de la ruta y se calcula el nuevo punto como la media de la posición calculada por odometría, la posición GNSS y la posición del punto de la ruta en el que se encuentra el vehículo.

Cada una de estas iteraciones se ejecuta en un tiempo de 100 milisegundos, lo que nos asegura una navegación y seguimiento casi en tiempo real.

La iteración termina cuando se alcanza el último punto del itinerario.

Para testear previamente la ruta que realizará el vehículo, se ha realizado un script análogo al anterior pero sin emplear el sensor GNSS.

7. PRUEBAS

En el siguiente apartado se exponen las pruebas realizadas, con el fin de testear el vehículo 2 y sus capacidades para la navegación autónoma. Se han realizado dos pruebas, la primera consiste en relazar un registro de datos GNSS en modo estático para analizar la exactitud y precisión del sensor GNSS utilizado. La segunda prueba consiste en establecer varios itinerarios y observar el comportamiento del ROVER 2, además de registrar los datos medidos mediante el sensor GNSS y los motores.

7.1 Pruebas GNSS estático

Se han realizado dos pruebas en dos lugares diferentes con tiempos de observación diferentes, con el fin de estimar la exactitud y precisión del sensor GNSS.

Las pruebas se han llevado a cabo en el exterior e interior de la pista de atletismo de la ciudad de Requena. La pista se encuentra al Sur-Oeste de la ciudad.

En las Figuras 38 y 39, se muestra la zona empleada para observar los dos puntos.

Figura 38: Zona pruebas GNSS estatico. Vista general.

Fuente: Visor SIGNA IGN

Figura 39: Zona pruebas GNSS estático. Vista detalle. Fuente: Visor SIGNA IGN

La primera observación se realizó en el exterior, sobre la esquina del bordillo de la isleta situada al norte de la pista de atletismo. El tiempo de observación fue de 4 minutos, en el cual se registraron 240 posiciones.

Las coordenadas del punto en el sistema ETRS89 WGS84 Huso 30 son:

X = 662427.63 metros

Y = 4372126.07 metros

En la Figura 40, se muestra el dibujo de las posiciones registradas.

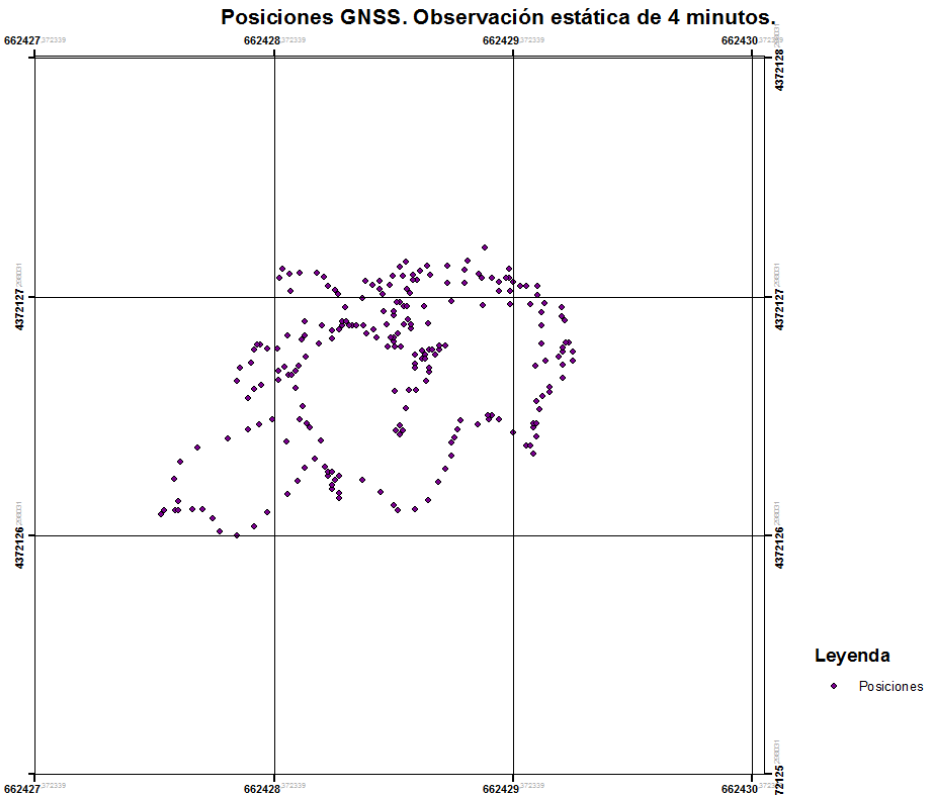

Figura 40: Posiciones registradas en la primera observación.

El punto medio calculado como la media aritmética de cada componente x e y de las observaciones realizadas es:

X = 662428,867

Y = 4372127,003

La diferencia en la componente X e Y del punto calculado a partir de la media y sus coordenadas reales es de 1,24 y 0,935 metros respectivamente.

El punto central calculado a partir de las observaciones y definido por la mediana es:

X = 662428,89

Y = 4372127,07

La diferencia en la componente X e Y del punto calculado a partir de la mediana y sus coordenadas reales es de -0,02 -0,07 metros respectivamente.

La segunda observación se realizó en el interior, sobre una esquina de la grada de la pista de atletismo. El tiempo de observación fue de 10 minutos en el cual se registraron 600 posiciones.

Las coordenadas del punto en el sistema ETRS89 WGS84 Huso 30 son:

X = 662463.92 metros

Y = 4372085.71 metros

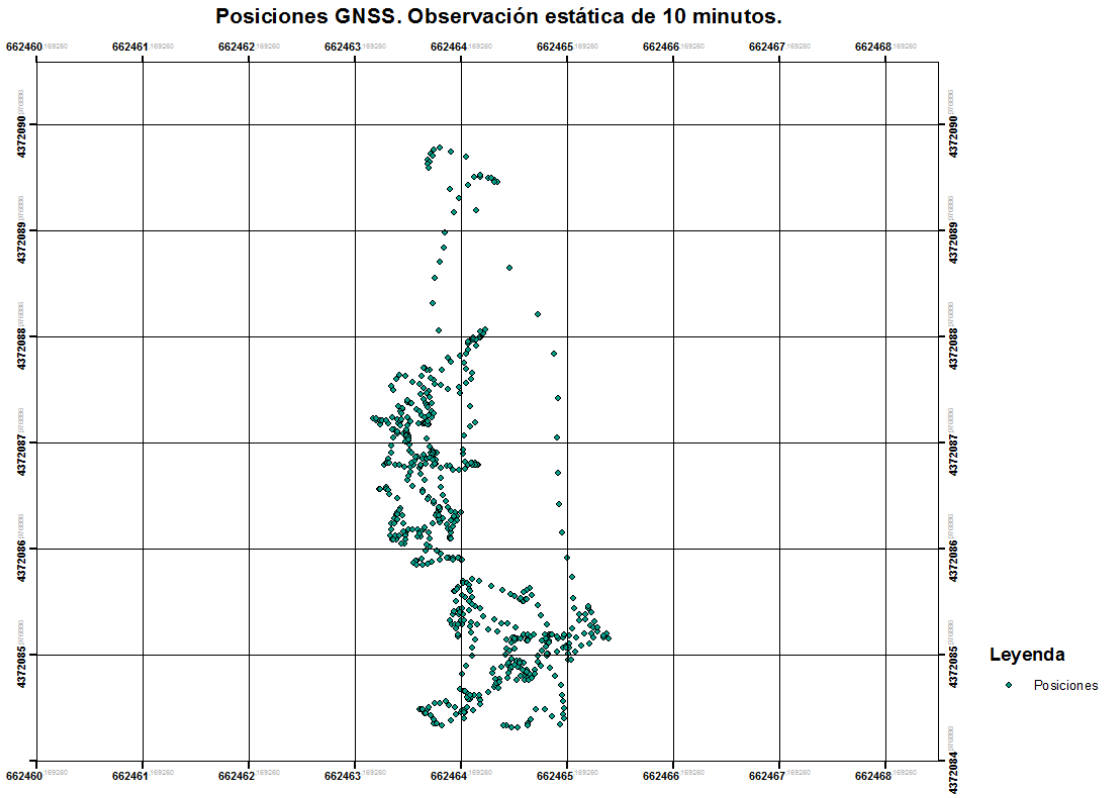

Figura 41: Posiciones registradas en la segunda observación.

El punto medio calculado como la media aritmética de cada componente x e y de las observaciones realizadas es :

X = 662464,21

Y = 4372087,16

La diferencia en la componente X e Y del punto calculado a partir de la media y sus coordenadas reales es de 0,29 y 1,45 metros respectivamente.

El punto central calculado a partir de las observaciones y definido por la mediana es:

X = 662464,121

Y = 4372087,128

La diferencia en la componente X e Y del punto calculado a partir de la mediana y sus coordenadas reales es de 0,09 y 0,03 metros respectivamente.

En las dos pruebas realizadas se puede observar la gran dispersión del sensor GNSS. Utilizar la posición registrada en un solo instante nos hace arriesgarnos a que las coordenadas de ese punto, en ese instante, difieran en exceso del valor real. En ambas representaciones se puede observar la mejora en el posicionamiento estático si se realizan observaciones durante un corto periodo de tiempo y se emplean mediadas estadísticas sobre estas.

7.2 Pruebas itinerarios

Se han realizado pruebas en dos zonas. La primera se corresponde con una explanada ubicada en el recinto ferial de Requena (Figura 42).

Figura 42: Zona de Pruebas, Recinto Ferial. Fuente: Visor IBERPIX

La trayectoria marcada consta de 20 puntos, realizando la forma que se muestra en la Figura 43:

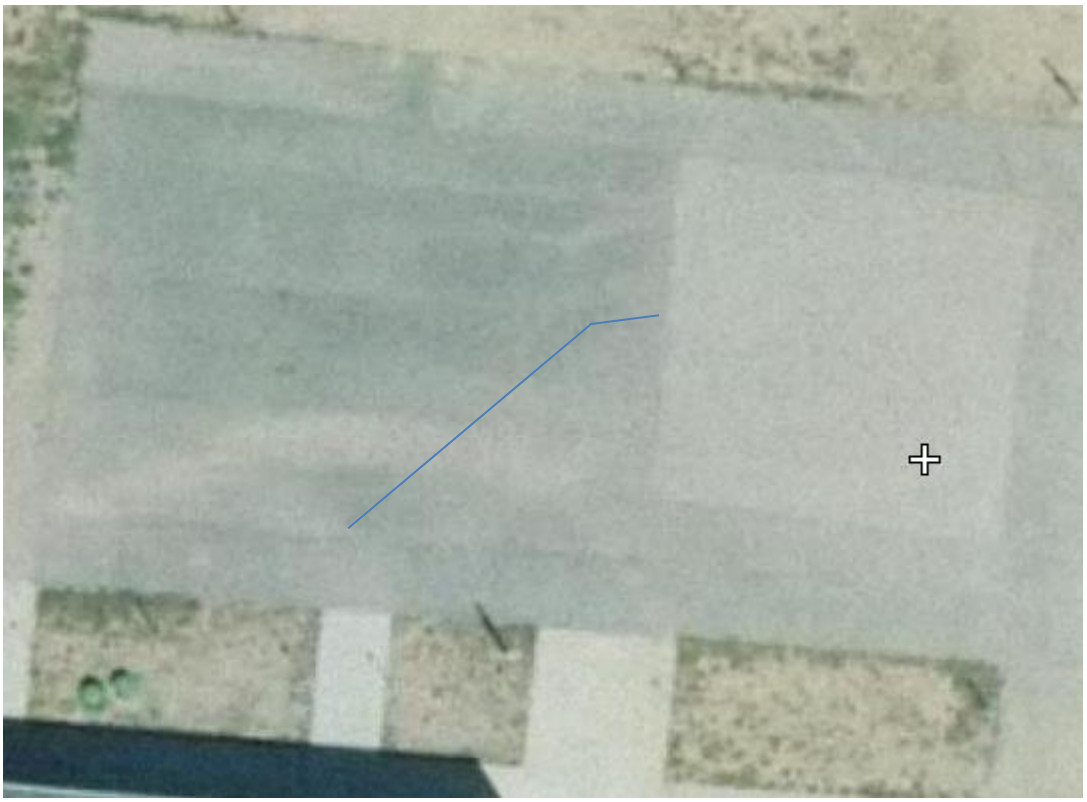

Figura 43: Trayectoria. Fuente: Visor IBERPIX

Para realizar este recorrido se ha empleado un tiempo de espera de 1 minuto desde la obtención de la primera posición GNSS hasta el arranque con el fin de sistema GNSS obtenga el mayor número de satélites posible .El vehículo se ha orientado de manera aproximada hacia el norte con la brújula incorporada en un smartphone con lo cual el ángulo respecto al eje x es de 90º. La distancia por debajo de la que se considera que se ha llegado al punto objetivo se ha fijado en 2.5 metros. Cuando llega a esa distancia se calcula la nueva posición como la media de la posición obtenida por GNSS, odometria y la posición del punto del itinerario en el que nos encontramos.

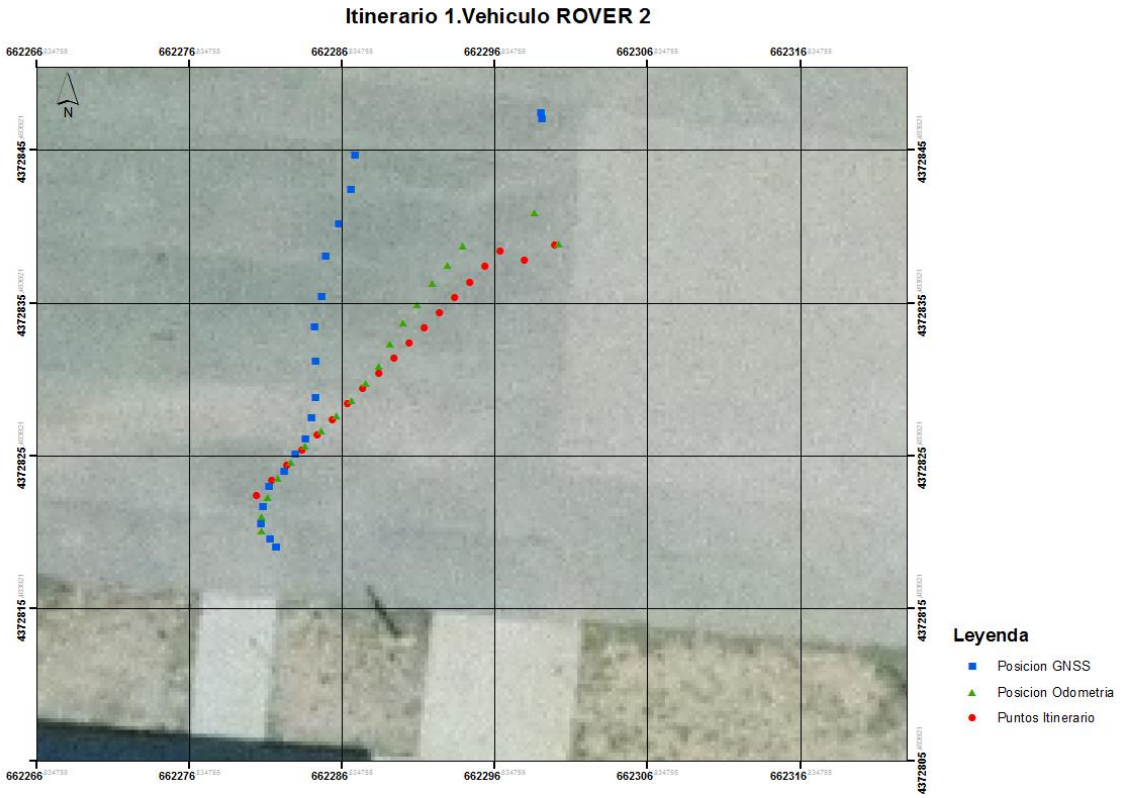

Figura 44: Itinerario 1 ROVER 2

Tabla 5: Tabla de resultados del itinerario del ROVER 2

	PUNTO ITINERARIO		GNSS		ODOMETRIA		Diferencia Itinerario-GNSS			Diferencia Itinerario-Odometria		
	X	Y	X	Y	X	Y	X	Y	Distancia	X	Y	Distancia
1	662280,24	4372821,88	662282,547	4372819,49	662282,053	4372820,21	-2,306732	2,39149	3,3226852	-1,812538	1,67469	2,46776834
2	662281,24	4372822,88	662282,547	4372819,49	662281,587	4372820,57	-1,306732	3,39149	3,634522382	-0,347122	2,30975	2,33568807
3	662282,24	4372823,88	662282,119	4372820,03	662281,597	4372821,48	0,120538	3,84524	3,847128808	0,643343	2,4008	2,48550415
4	662283,24	4372824,88	662281,553	4372821,04	662281,992	4372822,74	1,686583	3,8393	4,193421837	1,248365	2,14409	2,48103549
5	662284,24	4372825,88	662281,659	4372822,17	662282,657	4372823,99	2,581227	3,70814	4,518078694	1,582981	1,89353	2,46805282
6	662285,24	4372826,88	662282,047	4372823,51	662283,527	4372825,08	3,193407	3,36752	4,640909309	1,712814	1,79743	2,48283838
7	662286,24	4372827,88	662283,03	4372824,5	662284,449	4372826,16	3,210279	3,384529999	4,664861689	1,790809	1,71961	2,48275158
8	662287,24	4372828,88	662283,781	4372825,6	662285,511	4372827,1	3,459142	3,27693	4,764864489	1,729033	1,77802	2,48010287
9	662288,24	4372829,88	662284,434	4372826,62	662286,523	4372828,09	3,806343	3,26388	5,014096099	1,717166	1,78666	2,47806639
10	662289,24	4372830,88	662284,807	4372827,96	662287,511	4372829,1	4,432972	2,92357	5,3102262	1,728575	1,78139	2,48220101
11	662290,24	4372831,88	662285,079	4372829,33	662288,393	4372830,2	5,160653	2,5484	5,755578333	1,846741	1,67674	2,49437554
12	662291,24	4372832,88	662285,13	4372831,72	662289,261	4372831,37	6,110292	1,15996	6,219419227	1,979441	1,51092	2,49019395
13	662292,24	4372833,88	662285,026	4372833,92	662290,018	4372832,79	7,213783	-0,040250001	7,213895289	2,2225	1,08625	2,47375126
14	662293,24	4372834,88	662285,5	4372835,95	662290,872	4372834,19	7,740028	-1,06735	7,813275207	2,368274	0,68637	2,46573022
15	662294,24	4372835,88	662285,76	4372838,6	662291,786	4372835,41	8,480094	-2,71934	8,905436782	2,454019	0,46948	2,49852371
16	662295,24	4372836,88	662286,62	4372840,67	662292,752	4372836,79	8,620115	-3,79161	9,417148666	2,488242	0,08784	2,48979198
17	662296,24	4372837,88	662287,39	4372842,93	662293,752	4372838	8,850011	-5,047	10,18797839	2,488043	-0,12087	2,49097722
18	662297,24	4372838,88	662287,672	4372845,21	662294,788	4372839,22	9,568048	-6,32924	11,47200163	2,451512	-0,34381	2,47550326
19	662298,81	4372838,3	662299,831	4372847,98	662301,033	4372839,38	-1,021295	-9,681	9,734721592	-2,223302	-1,08199	2,47260473
20	662300,81	4372839,3	662299,912	4372847,58	662299,476	4372841,41	0,898451	-8,2755	8,32412845	1,333908	-2,11064	2,49682033

En la Figura 44 podemos ver los puntos que se registran cuando el vehículo está a 2.5 metros de un punto del itinerario. Además, en la misma figura y en la Tabla 5, se puede observar como a media que aumenta la distancia los errores del método se van acumulando haciendo que el vehículo se desvíe de la trayectoria que debe seguir.

La estimación de la posición por odometría se vuelve una técnica imprecisa para el vehículo en distancias a partir de los 15 metros. Los pequeños errores de calibración del vehículo, la pequeña diferencia en la orientación inicial respecto a la orientación de referencia, junto con las irregularidades del suelo provocan que a medida que el vehículo avanza este valla acumulando errores que aumentan con el transcurso del avance.

La estimación del vehículo por odometría considera que no tiene error en el cálculo de su posicionamiento cuando en realidad si lo tiene. Sin la posición

ofrecida por el GNSS el vehículo no sería consciente de que se desvía de la trayectoria.

Así es como entra en juego la posición por satélite, para posicionar el vehículo correctamente. Pero esta posición tampoco está exenta de errores lo que también introduce una cantidad de error en la posición del vehículo.

Así es como surge la idea de calcular el punto medio formado por el punto del itinerario, el punto GNSS y el odométrico llamado Baricentro, por ser el centro del triángulo formado por los tres puntos.

Utilizando esta técnica se intenta suavizar y compensar los errores de ambos posicionamientos.

PRUEBA EN ZONA PISTA DE ATLETISMO

En esta zona se elaboraron pruebas empleando la siguiente estrategia:

1. El vehículo inicia el recorrido orientado al norte.

2. Para mejorar su posicionamiento inicial el vehículo permanece parado obteniendo lecturas GNSS durante 1 minuto, obtenidas estas coordenadas se calcula la mediana de estas posiciones y el valor resultante será el punto de partida del vehículo hacia el itinerario.

Con el fin de mejorar la orientación del vehículo, cada vez que el vehículo alcance un punto objetivo del itinerario, la nueva posición del vehículo será la obtenida únicamente GNSS y la orientación será la recalculada utilizando el azimut calculado por el sensor GNSS. El inconveniente de este método es que en ocasiones el azimut no es ofrecido por el sensor GNSS en cuyo caso se sigue utilizando el odométrico.

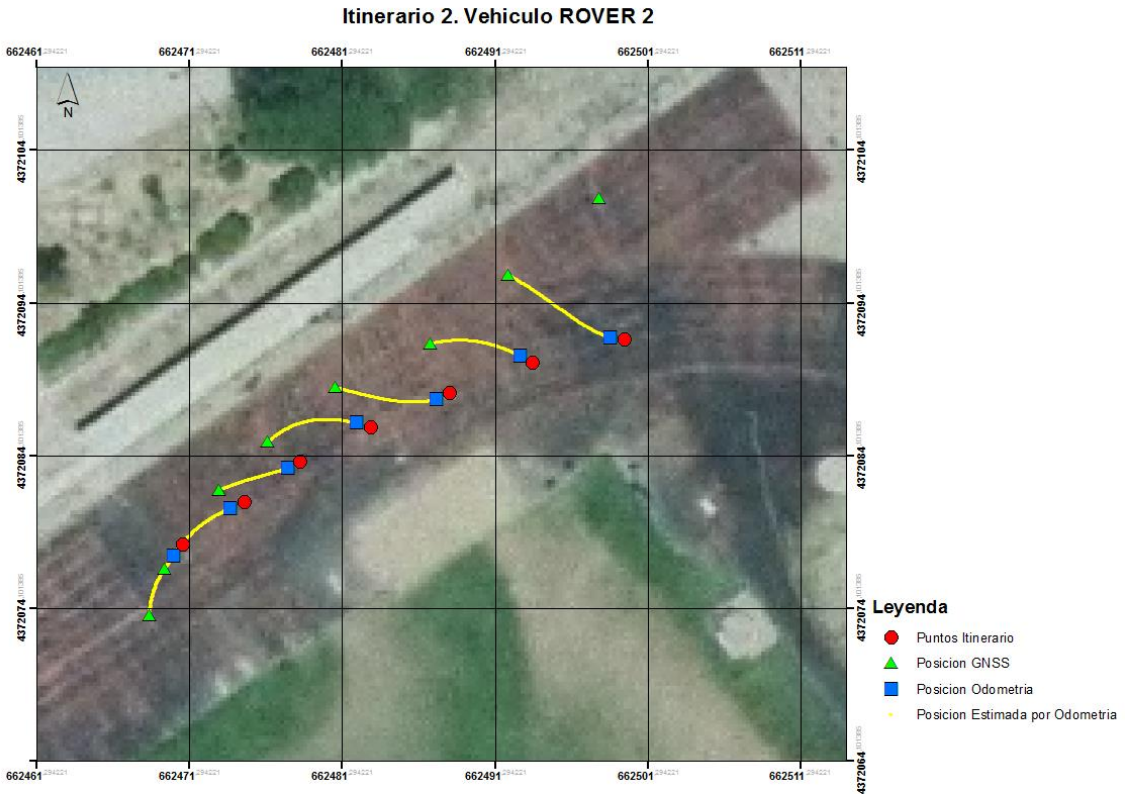

Figura 45: Itinerario 2 ROVER 2.

En la Figura 45 se puede observar como la trayectoria no se pudo corregir en ningún punto al no disponer del azimut por GNSS. Esto se traduce en que el rover corrige su posición pero no la trayectoria lo cual provoca que el vehículo se valla desviando de la trayectoria.

La disponibilidad o ausencia aleatoria del azimut por parte del sensor GNSS es un fenómeno anormal que en un receptor de mayor calidad no se produciría.

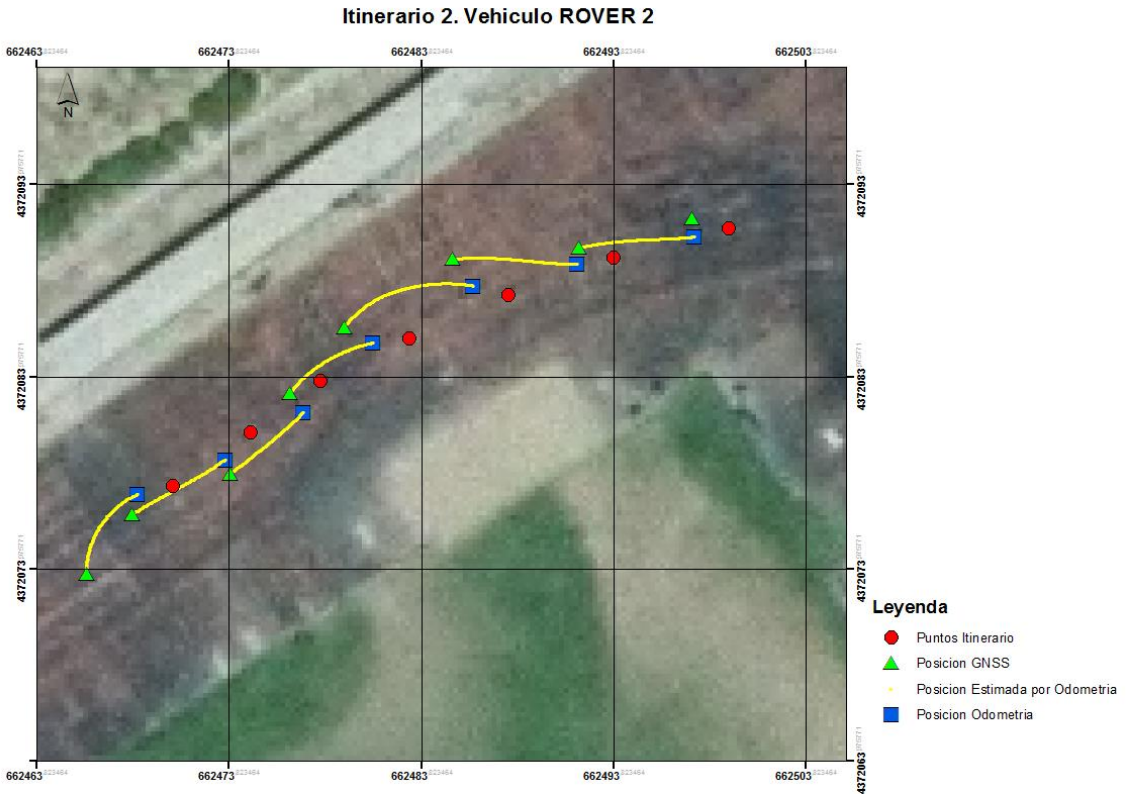

Figura 46: Itinerario 2 ROVER 2.

El la Figura 46 se muestra el mismo itinerario realizado en el que si se aplicaron correcciones en la trayectoria mediante GNSS y se puede apreciar como mejora la navegación del vehículo.

8. MEJORAS

En este apartado, se comentarán las posibles mejoras a introducir en el vehículo ROVER 2 para mejorar y completar las funcionalidades de las que dispone en la actualidad. A continuación, se describen algunas de ellas.

BRÚJULA:

Uno de los principales inconvenientes del ROVER 2 es la orientación manual previa al inicio del programa para recorrer el itinerario. Acoplando al sistema una brújula, esta orientación seria conocida y el vehículo sería capaz de auto orientarse incluso en caso de no poder recuperar la orientación del receptor GNSS.

GNSS DIFERENCIAL DE FASE:

En la actualidad, el vehículo ROVER 2 hace uso de un sensor GNSS de solución de código, el cual ofrece una precisión por debajo de lo aceptable para navegación de precisión. Para futuras mejoras, se prevé sustituir el receptor GNSS por uno de solución de fase (RTK). Esto nos proporcionará un mejor posicionamiento, lo que se traducirá en una navegación más exacta.

UNIDAD INERCIAL IMU:

Para futuras modificaciones y mejoras, sería interesante incluir en el sistema una unidad inercial para conocer las variaciones en aceleraciones y giros, con el fin de mejorar la navegación.

9. CONCLUSIONES

Para finalizar, se exponen las conclusiones extraídas tras la realización del presente trabajo. Previamente, cabe reseñar que se han cumplido todos los objetivos descritos al inicio del trabajo.

En la realización del presente trabajo de fin de master, se han adquirido las nociones necesarias sobre robótica y electrónica y se han aplicado conocimientos referentes a las tecnologías de posicionamiento espacial.

Las conclusiones son:

Se han desarrollado de manera satisfactoria dos vehículos operativos que utilizan Sistemas Globales de Navegación por Satélite, o GNSS, como método para el posicionamiento o geolocalización en su recorrido de manera autónoma, de itinerarios en el mundo real.

Se ha desarrollado un sistema de navegación manual para controlar remotamente el desplazamiento de los vehículos con una comunicación basada en redes WIFI.

Se han experimentado varias estrategias para la operación autónoma del vehículo, en concreto odometría y posicionamiento por satélite, así como la combinación de ambas tecnologías.

Se ha comprobado que el receptor GNSS empleado no es adecuado para ciertas aplicaciones que requieran navegación de precisión. Esto es un factor limitante en ciertas aplicaciones.

Los ordenadores de placa simple (SBC) y la programación Python constituyen un entorno de trabajo adecuado y un punto de partida interesante para la realización de vehículos autónomos para aplicaciones que requieran la recolección de datos sistemática en exteriores.

Aunque ha sido necesario adquirir nuevos conocimientos, gracias a los contenidos estudiados en la titulación del Máster en Geomática y Geoinformación se ha conseguido desarrollar un trabajo en que la pieza clave es la Geomática, sin la cual todo el trabajo carecería de sentido.

BIBLIOGRAFÍA

[1] Berné Valero, J. L. (2013). Apuntes Geodesia espacial tema 4 GPS.

[2] Donat, W. (2015). *Make a raspberry Pi–Controlled robot.* Sebastopol: Maker Media.

[3] Lundgren, M. (2003). Path tracking and obstacle avoidance for a miniature robot.

[4] *Pagina fundación Raspberry pi.*, 2016, from https://www.raspberrypi.org/

[5] Payne, P. (2013). *A python library for controlling the MD25 motor controller on an i2c bus.*, 2016, from https://github.com/payneio/md25_motor_controller_library

[6] Whitaker, J. (2016). *Python interface to PROJ.4 library.,* 2016, from https://pypi.python.org/pypi/pyproj?