UNIVERSIDAD POLITÉCNICA DE CATALUÑA

FACULTAD DE INFORMÁTICA DE BARCELONA

IMPLEMENTACIÓN DEL SISTEMA DE LOCALIZACIÓN Y OBTENCIÓN DEL MAPA DE UN ROBOT MÓVIL

RAMÓN JAIME ARANDES MIRALLES

PROYECTO FINAL DE CARRERA EN INGENIERÍA INFORMÁTICA

DIRIGIDO POR ANTONIO-BENITO MARTÍNEZ VELASCO

Octubre 2015

Información del Proyecto

Título: **Implementación del sistema de localización y obtención del mapa de un robot móvil**

Autor: **Ramón Jaime Arandes Miralles**

Fecha: **26 de Octubre de 2015**

Titulación: **Ingeniería Informática (Plan 2003)**

Créditos: **37,5**

Director: **Antonio-Benito Martínez Velasco**

Departamento: **Enginyeria de Sistemes, Automática i Informática Industrial**

Tribunal

Presidente: **Josep Fernández Ruzafa**

Firma:

Secretario: **Antonio-Benito Martínez Velasco**

Firma:

Vocal: **Manel Guerrero Zapata**

Firma:

Cualificación

Cualificación numérica:

Cualificación descriptiva:

Fecha:

Resumen

Este proyecto abarca todo el proceso de configuración de un robot comercial (Neato XV-Essential) para poder controlarlo de forma remota mediante conexión inalámbrica, la implementación de diversos entornos de control para su uso en docencia (Matlab y Simulink), y la implementación de otro entorno específico en web (HTML y Javascript), para desarrollar una funcionalidad sencilla como es la extracción de puntos de interés del entorno ('*Landmarks*'). Este proceso de extracción de puntos de interés es fundamental a la hora de implementar un sistema de localización autómatico, ya que es el paso previo a la corrección de la localización.

Agradecimientos

A mi familia, por su apoyo incondicional.

Al profesor Antonio Martínez, por transmitirme sus conocimientos y permitirme realizar el proyecto en el departamento de ESAII convirtiéndose en una experiencia tan interesante cómo enriquecedora.

Al departamento de ESAII, por su ayuda en todo aquello que he necesitado.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El proyecto surge de una propuesta personal al Profesor Antonio Martínez, la cual se convierte, gracias a su visión, en un proyecto aplicable a la docencia.

La idea de realizar un proyecto sobre robótica, y en este caso, de robótica móvil proviene del interés personal en este campo. Siempre me ha interesado dentro del mundo de la informática, la rama dedicada al control. Aquella que une la parte intangible de la informática, cómo el código, con el mundo real mediante hardware, sensores y actuadores.

La motivación también proviene del auge en los últimos años de la investigación en el campo de la robótica, con el fin de incorporarla paulatinamente en el día al día de la sociedad. Se habla en un futuro próximo de vehículos autónomos y de robots humanoides de asistencia a las personas, tanto en el ámbito clínico como en el hogar.

Por tanto se trata de un tema muy actual y con una clara proyección de crecimiento en el fututo.

1.2. Objetivos

El principal objetivo de este proyecto es la obtención de una plataforma robótica con la que poder realizar prácticas de laboratorio para las asignaturas de Robótica en los estudios de Grado y de Máster, y también proyectos finales de estudiantes.

Esta plataforma tiene que poder ser contralada de forma remota mediante conexión inalámbrica aprovechando la gran expansión de éstas. Siendo capaces de enviar comandos de movimiento y obtención de datos del robot cómo del entorno, para proyectos de localización y obtención del mapa, mediante un software que realiza una conexión por socket con el robot implementado en cualquier lenguaje de programación.

Una vez conseguidas estas funcionalidades, también se quiere demostrar la gran versatilidad del proyecto, implementando una interfaz web con la que controlar el robot desarrollando alguna funcionalidad sencilla demostrando sus funciones.

Capítulo 2

Estado del arte

2.1. Robótica Móvil

Los robots móviles, tienen la capacidad de desplazarse alrededor de un determinado entorno y no están fijos en una posición determinada. En contraposición, otros robots, como los industriales, generalmente consisten en un brazo articulado y una herramienta que está anclada a una superficie fija.

Los robots móviles son foco de investigación actual, y casi todas las grandes universidades tienen un laboratorio dedicado al desarrollo de aplicaciones en robótica móvil. Los robots móviles también pueden ser encontrados en la industria y en entornos militares y de seguridad.

También posible encontrarlos en entornos domésticos, tanto como juguetes o realizando tareas domésticas sencillas.

Para que los robots móviles puedan cumplir su cometido, deben conocer dónde se encuentran, a dónde ir y que acciones realizar para llegar a su objetivo. Para ello, se deben cumplir cuatro requisitos:

Control: La capacidad de accionar los actuadores del robot para llevar a cabo los movimientos deseados.

Percepción: La obtención de datos, tanto del propio robot como del entorno. Estos pueden provenir de todo tipo de sensores, como encoders, LIDARs o cámaras.

Localización: La capacidad de situar al robot en su entorno, estimando su posición y orientación

(pose) en cada momento.

Planificación: Tarea de decisión sobre qué acciones debe realizar el robot para llevar a cabo sus objetivos.

En este proyecto, se aplican tanto el control, cómo la percepción y la localización. El control se lleva a cabo mediante el envío de comandos a través de la solución que se ha implementado. La percepción y la localización se detallan a continuación, ya que requieren más explicación.

2.2. Percepción

La capacidad de percepción vendrá definida por el tipo de sensores que incorpore el robot móvil y que le permitirá obtener información del entorno en que se encuentra. Se detallan los sensores más comunes:

2.2.1. Encoders

Estos sensores cuentan tics de rueda (fracciones de vuelta). Sabiendo el número de tics transcurridos en una fracción de tiempo y el radio de la rueda, se puede calcular la velocidad angular de giro. Existen diferentes tipos de encoders según la tecnología de contabilización de tics, los más comunes son ópticos o magnéticos.

Es el sensor más común para estimar la posición mediante la odometría.

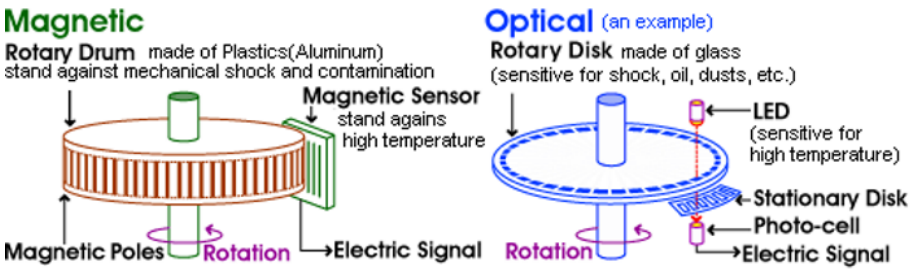

Figura 2.1: Detalle de funcionamiento de encoders magnéticos y ópticos.

2.2.2. Visión

Una cosa que falta a todos los sensores de detección de distancias es la habilidad de extraer propiedades de las superficies observadas e identificar objetos. Las imágenes a color (o escala de grises)

nos permiten utilizar un amplio repertorio de información para identificar y localizar componentes del entorno.

Las principales ventajas de los sensores de visión son:

- Gran cantidad de información.

- Capacidad de extraer información en tres dimensiones del entorno.

- Las cámaras son sensores pasivos, que no necesitan emitir sonido o luz cómo los sonar y los escáneres laser.

2.2.3. Láser

El escáner láser es el sensor que está en continua expansión en mundo de la robótica. Las técnicas dominantes para la obtención de medidas de distancia basadas en laser son las denominadas técnicas de tiempo de vuelo (TOF, por sus siglas en inglés) y las técnicas de cambio de fase.

Un sistema TOF lanza un pulso laser de corta duración y se mide el tiempo que tarda este en volver. La distancia se calcula entonces como un medio de la velocidad de la luz por el tiempo que ha tardado el láser en realizar un viaje de ida y vuelta.

En sistemas de cambio de fase, se transmite una ola continua de luz. La idea es comparar la fase la señal retornada con una señal de referencia generada por la misma fuente.

Un escáner laser presenta multitud de ventajas:

- Es rápido. El tiempo de medida puede considerarse instantáneo.

- La precisión es bastante elevada. Las nuevas generaciones de lasers tienen una precisión con un margen de error inferior a 10 milímetros.

- La resolución angular de 0.5º o de 0.25º dependiendo del fabricante, es superior a la ofrecida por un sónar.

- Los datos obtenidos de un escáner láser pueden ser interpretados directamente como la distancia hasta un obstáculo en una posición determinada.

El escáner láser es el mejor sensor para extraer propiedades de superficies planas, como las paredes, gracias a la densidad de la información de distancias que ofrece.

Figura 2.2: Sensor laser de 360º de rotación RPLidar.

2.2.4. Sonar

Este tipo de sensores funcionan tanto como emisores como receptores de ultrasonidos, de forma análoga a los sensores laser. Igual que estos, utiliza técnicas TOF para detectar distancias.

El éxito obtenido al aplicar este tipo de tecnologías en robots móviles depende en gran medida del acercamiento utilizado en la clasificación de los datos obtenidos.

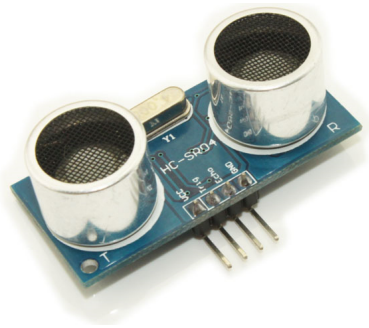

Figura 2.3: Sensor de distancias por sonar.

2.3. Localización

2.3.1. Odometría

El concepto de odometría se define como el estudio de la estimación de la posición de vehículos con ruedas durante su navegación. En robots móviles se utiliza para estimar su posición relativa a su localización inicial. Es bien sabido que dada una buena aproximación respecto a su localización inicial, la odometría proporciona una buena precisión a corto plazo, no obstante, la odometría es la integración

de información incremental del movimiento a lo largo del tiempo, lo cual conlleva inevitablemente a la acumulación de errores.

La odometría se basa en ecuaciones sencillas de implementar, las cuales hacen uso del valor de los encoders de las ruedas del robot para traducir las revoluciones de las ruedas a un desplazamiento lineal relativo al suelo. Este concepto básico puede llevar a imperfecciones y a cálculos erróneos si por ejemplo las ruedas patinan, o se produce una sobre-aceleración (errores no sistemáticos), también depende directamente de la exactitud de la localización y orientación inicial del robot.

Situar un robot a mano, en una posición determinada con una precisión de milímetros, puede no resultar excesivamente complicado, pero cuando se trabaja con varios robots los cuales deben compartir las coordenadas absolutas de un escenario, puede resultar realmente complejo e imperfecto. Además, el correcto funcionamiento de la odometría depende también de errores sistemáticos como a medición de los diámetros de las ruedas, su alineamiento, la resolución discreta del encoder o la distancia de separación entre las ruedas.

A pesar de sus limitaciones, el uso de la odometría en investigación es uno de los pilares más importantes del sistema de navegación de un robot.

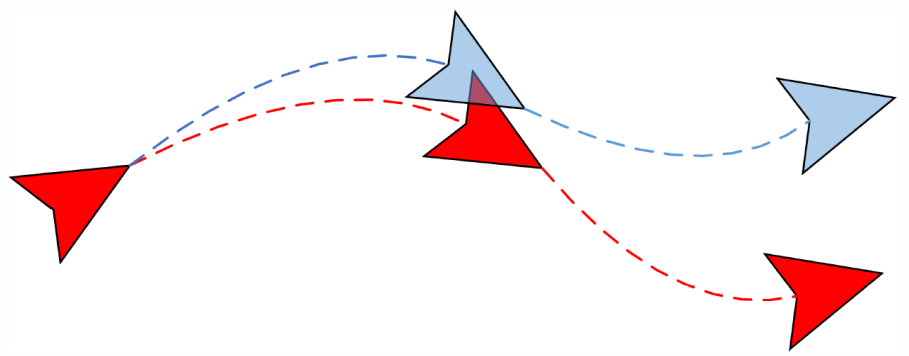

Figura 2.4: Comparación entre la posición real (azul) y la estimada por odometría (rojo).

2.3.2. Extracción de puntos de interés (Landmarks)

Una opción para solucionar el problema de la odometría es utilizar un mapa y características del entorno para corregir la estimación. Por ejemplo, esto es lo que llevan haciendo los barcos durante siglos gracias a los faros. Conociendo su localización en un mapa y mediante observaciones se puede ir corrigiendo la estimación hecha hasta entonces.

Hoy en día tenemos métodos más sofisticados como el GPS, capaz de darnos una precisión de metros siempre que tengamos cobertura. Sin embargo, este sistema no funciona si necesitamos más precisión o si nos encontramos en interiores, dónde las señales del GPS no están disponibles.

No siempre es posible recurrir a marcas conocidas para usar como referencia. En ese caso, deberá ser el observador el que decida qué elementos del entorno deberán usarse como referencias. Esta es una elección muy importante, ya que de ello dependerá el éxito de la localización. Las estrellas o accidentes geográficos son ejemplos de características naturales que se llevan utilizando desde la prehistoria.

Confundir una característica por otra puede resultar desastroso. Supondrá corregir la posición en base a información errónea, es decir, empeorarla.

En el caso de un robot, esas características deberán de ser extraída de los datos proporcionados por los sensores. Dicho proceso de extracción se conoce por su nombre en inglés *feature extraction*, es decir, extracción de características.

El tipo de sensores utilizados limitará el tipo de características que se pueden extraer. Por ejemplo, si se utiliza una cámara, el tipo de información que se podrá extraer será muy alto, aunque también su coste computacional. Si se utilizara un simple sensor de distancia, la información será muy limitada.

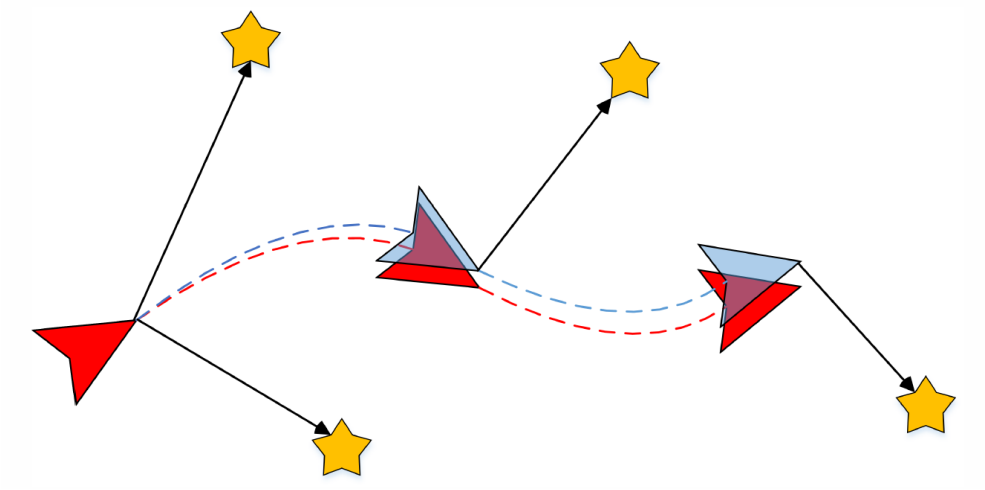

Figura 2.5: Usando elementos del entorno se puede corregir la estimación de la posición.

Capítulo 3

Entorno de Trabajo

3.1. Neato XV-Essential

Se trata de un robot comercial de limpieza de la marca Neato. Adquirimos el modelo XV-Essential ya que incorporaba todos los sensores y actuadores necesarios a un precio asequible.

Lo que diferencia a este robot de otras marcas es su sensor laser de 360º de rotación, el cual es muy interesante para proyectos de localización y obtención del mapa.

Figura 3.1: Neato XV-Essential

A continuación se detallan las características más importantes:

- Sensor de paredes en el lateral frontal derecho
- 4 bumpers, dos en el frontal y uno en cada lateral
- 2 sensores de caída en el frontal
- Laser rotatorio de 360º
- Encoders con 1mm. de resolución

- Acelerómetro

- Puerto USB de comunicación

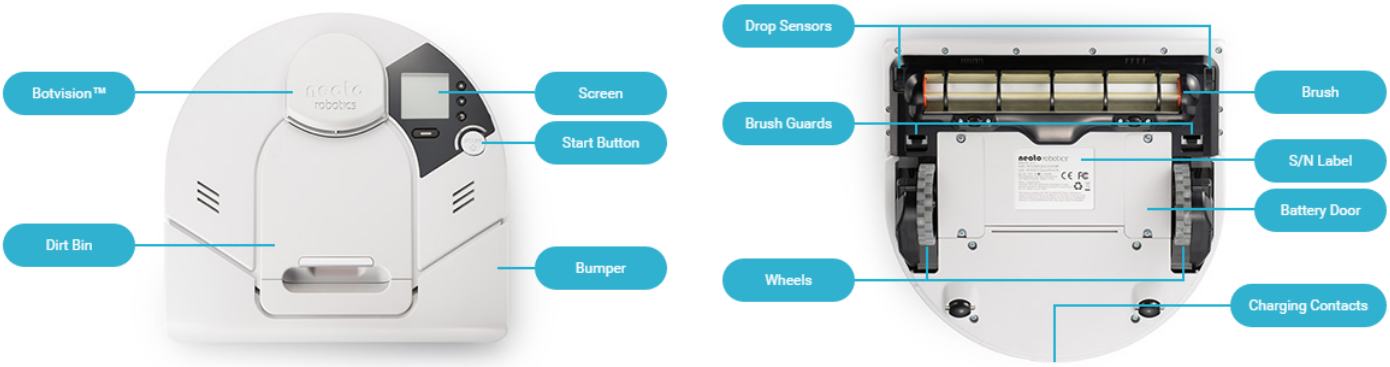

Figura 3.2: Detalle de los sensores y actuadores

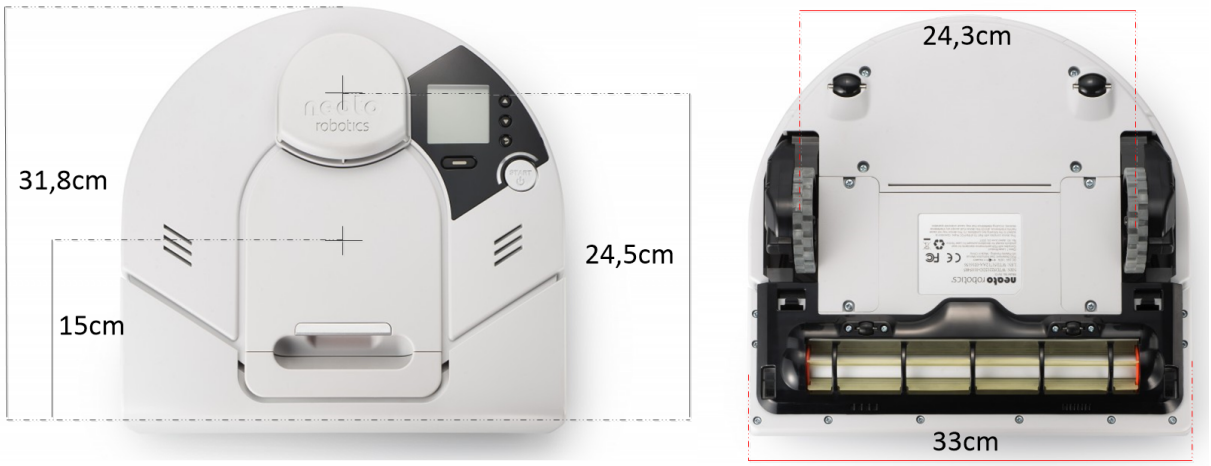

Figura 3.3: Detalle de las medidas

3.1.1. Puerto serie

El puerto USB que incorpora el robot Neato implementa un puerto de comunicación serie genérico. Éste acepta diferentes tipos de comandos tanto de control como de obtención de datos. Los comandos que acepta el robot se encuentran en el anexoA.

3.2. Raspberry Pi 2 Model B

Es un ordenador de placa reducida o placa única (Single Board Computer) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Este modelo es la segunda generación, reemplaza el original Raspberry Pi 1 Model B+. Sus principales características son:

- Procesador de 4 núcleos ARM Cortex-A7 a 900 Mhz
- 1Gb de RAM
- 4 puertos USB
- 40 puertos GPIO
- Puerto full HDMI
- Puerto ethernet
- Puerto combinado de jack de audio y video compuesto
- Interfaz de cámara (CSI)
- Interfaz de display (DSI)
- Slot de tarjeta SD
- Núcleo de video VideoCore IV 3D

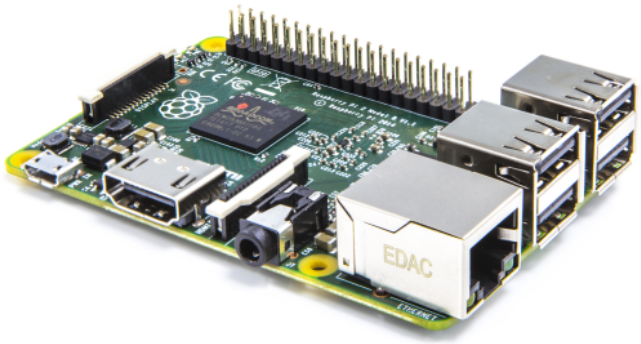

Figura 3.4: Raspberry Pi 2 Model B

Para la realización de este proyecto se le ha añadido una llave USB Wifi para poder disponer de conexión inalámbrica. El modelo es un TP-Link Nano USB Adapter TL-WN725N.

3.3. Matlab

El entorno escogido para desarrollar las interfaces de usuario para finalidad docente ha sido Matlab. Es un entorno muy amigable a la hora de trabajar con modelos ya que simplifica mucho la manipulación y visualización de datos.

Está enfocado en el manejo de matrices y vectores (de ahí su nombre **MAT**trix **LAB**oratory), siendo muy eficiente a la hora de realizar cálculos con ellos.

Está disponible para los principales sistemas operativos: Windows, GNU/Linux y MacOS. Cada año se lanzan dos versiones, identificadas por el año y la letra 'a' o 'b' según sea la primera o segunda del año.
La versión utilizada ha sido la R2014a.

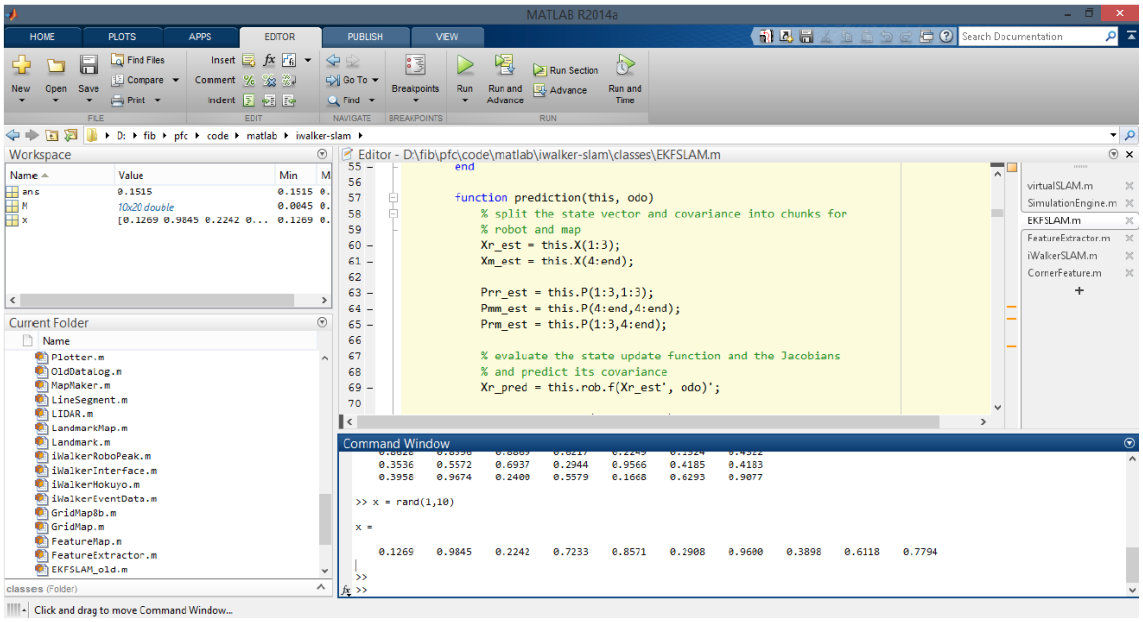

Figura 3.5: Interfaz Matlab

3.3.1. Lenguaje de programación M

Matlab dispone de su propio lenguaje de programación llamado M. Se trata de un lenguaje interpretado y de tipo dinámico.

Originalmente estaba limitado al uso de scripts y funciones, pero actualmente dispone de programación

orientada a objetos. Permite también la creación de interfaces de usuario sencillas, funcionalidad que se ha usado para crear la interfaz del proyecto para fines docentes.

Un inconveniente de este lenguaje es que no permite ejecutar más de un hilo de ejecución de forma explícita.

3.3.2. Simulink

Es un entorno de programación visual que funciona sobre el entorno de programación Matlab, diseñado específicamente para la creación de modelos, simulaciones y sistemas dinámicos. Permite la realización de modelos tratando datos tanto discretos como continuos.

Su interfaz principal es una herramienta gráfica de creación de diagramas de bloques. Ofrece integración con el resto del entorno Matlab y puede manejar su entorno o ejecutar scripts procedentes del mismo. Es bastamente usado en control automático y en proceso de señales digitales en simulaciones y diseño basado en modelos.

Se ha usado para crear un modelo básico de control para el robot, de forma que se tiene un objeto que representa al robot, mediante el cual, indicándole una entrada para realizar su control, se obtiene una salida con los datos del mismo. De esta forma se dispone de otro entorno, en este caso, en forma de modelo, para poder realizar el control del robot y cuyas implementaciones son muy amplias.

3.3.3. C++ y Java

Matlab permite ejecutar código C/C++ mediante lo que se conoce como funciones MEX. Se ha usado código C++ para diseñar la funcionalidad de una de las cajas principales del modelo del robot para Simulink. Y se ha usado Java, que se puede usar directamente sobre código M, para implementar la comunicación por socket con el robot en código M.

3.4. Código Python

Es un lenguaje de programación interpretado de tipado dinámico y multiplataforma, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Es administrador por la “Python Software Foundation”. Posee licencia de código abierto, denominada “Python Software Foundation License” que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1.
Se ha usado para programar la interfaz de comunicación entre el robot y el puerto socket.

3.5. Codigo HTML y Javascript

HTML, del inglés “HyperText Markup Language”, hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de paginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágaenes o videos, entre otros.

Es un estándar a cargo de la W3C “Wolrd Wide Web Consortium” organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web.

Javascript, abreviado comúnmente “JS”, es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objectos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado cliente, implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Todos los navegadores actuales interpretan el código Javascript integrado en las páginas web. Para interactuar con una página web, se provee al lenguaje Javascript de una implementación del "Document Object Model"(DOM).

Se han usado para desarrollar la interfaz de usuario específica para demostrar funcionalidades sencillas del robot. También se ha usado JQuery, que es una libreria de Javascript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos y desarrollar animaciones.

3.6. Arquitectura

Cómo último punto del entorno de trabajo, mostramos un diagrama de la arquitectura física implementada.

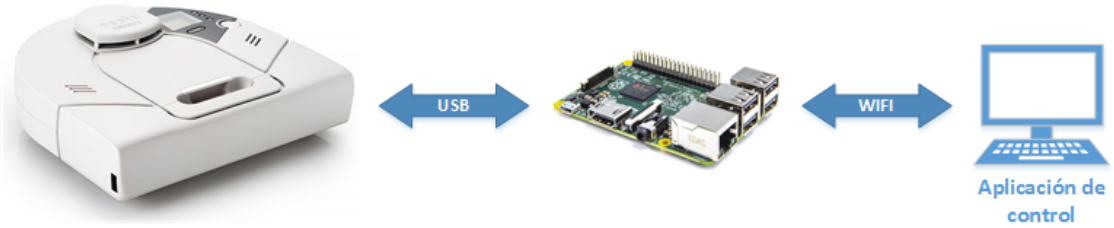

Figura 3.6: Esquema hardware

Capítulo 4

Cinemática del robot: Ecuaciones de movimiento

El robot Neato XV-Essential corresponde perfectamente a una modelización clásica de robot diferencial que explicaremos a continuación.

4.1. Robot diferencial

Existen varios modelos para representar a robots móviles según las características de éstos (número de ruedas, geometría...). El modelo del robot diferencial es el que mejor se ajusta a este robot. Como se puede apreciar en la figura, éste consiste en dos ruedas de radio r situadas en paralelo en el mismo eje una distancia *S*. Estas dos ruedas corresponden a las dos ruedas motrices del robot. El resto de elementos que entran en contacto con el suelo son rodamientos de contacto que no afectan al modelo, por lo que se obvian en la representación.

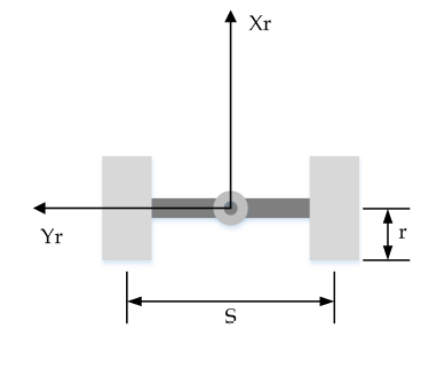

Figura 4.1: Modelo del robot diferencial

El eje de coordenadas local del robot, denotado $\{R\}$, está situado en el centro del eje motriz de las ruedas. El eje X_r apunta hacia la parte delantera del robot, y el eje Y_r hacia su lado izquierdo, siendo paralelo al eje que une las ruedas. El eje Z_r apunta hacia la perpendicular al plano que forman los ejes X_r y Y_r y en dirección contraria al suelo, aunque como vamos a trabajar en 2D, esta dimensión no se va a usar.

El estado o pose del robot se representa respecto a un sistema de coordenadas global, denotado $\{W\}$, como $X = (x, y, \theta)$.

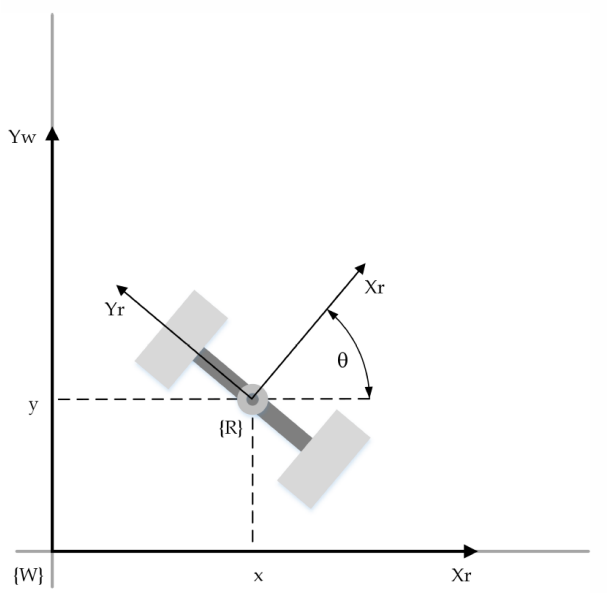

Figura 4.2: Sistema de coordenadas global W y sistema de coordenadas local R

Cada una de las ruedas gira de forma independiente respecto al eje Y_r al tiempo que está fijada respecto al eje Z_r. La velocidad de cada rueda se define cómo:

$$v_{\mathrm{l}} = \omega_{\mathrm{l}} r$$

$$v_{\mathrm{r}} = \omega_{\mathrm{r}} r$$

dónde $\omega_{\mathrm{l}} r$ y $\omega_{\mathrm{r}} r$ son las velocidades angulares izquierda y derecha respectivamente, expresada en radianes por segundo, r es el radio de las ruedas, expresado en metros, y v_l y v_r son las velocidades lineales, expresadas en metros por segundo.

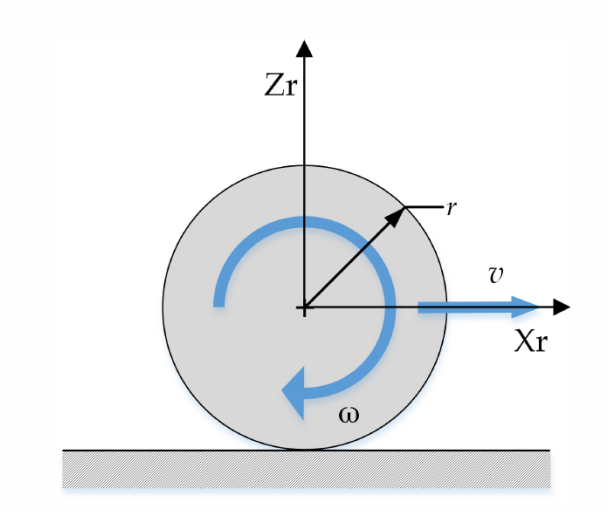

Figura 4.3: Rueda girando sobre el eje Y del robot

Cómo se puede apreciar, las velocidades de las ruedas siempre son paralelas al eje X_r. Mediante estas velocidades se puede definir la velocidad lineal v del robot, así cómo su velocidad angular ω.

$$v = \frac{v_r + v_l}{2}$$
$$\omega = \frac{v_r - v_l}{S}$$

Analizando las ecuaciones se pueden extraer los tres tipos de movimientos que puede realizar el robot: avanzar, girar y rotar. En la figura 4.4 se pueden ver los tres casos en detalle. Existen otros tres casos totalmente simétricos permitiendo retroceder y girar o rotar hacia la izquierda.

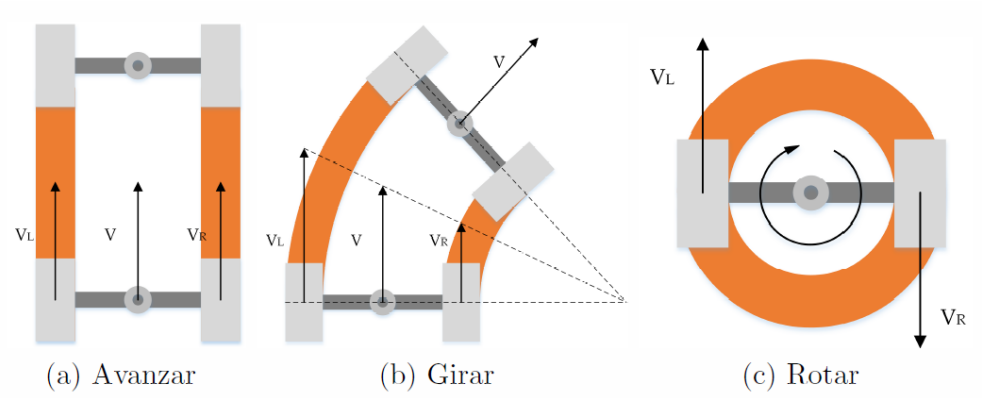

Figura 4.4: Movimientos posibles de un robot diferencial en función de las velocidades de sus ruedas.

A partir de la velocidad lineal y angular del robot, se puede calcular la odometría $\delta = (\delta_d, \delta_\theta)$ para un

intervalo k. Para que la odometría sea fiable, el intervalo de tiempo empleado debe ser lo más pequeño posible.

$$v = \frac{v_r + v_l}{2}$$
$$\omega = \frac{v_r + v_l}{S}$$

4.2. Cinemática directa

La cinemática directa consiste en determinar cuál es la posición y orientación del robot respecto a un sistema de coordenadas que se toma como referencia.

En el modelo del robot diferencial, la cinemática directa se puede extraer a partir del radio y la separación de las ruedas y sus velocidades angulares:

$$\begin{pmatrix} v(t) \\ \dot{\theta} \end{pmatrix} = \frac{r_w}{2S} \begin{pmatrix} S & S \\ 1 & -1 \end{pmatrix} \begin{pmatrix} w_R \\ w_L \end{pmatrix}$$

4.3. Trayectoria

Para conocer la trayectoria sólo hay que calcular integrar la sucesión de posiciones que se van produciendo. Conociendo todos los parámetros anteriores:

$$\begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} = \begin{pmatrix} x_{ini} + \int_0^t v(t) cos\theta \, dt \\ y_{ini} + \int_0^t v(t) sin\theta \, dt \\ \theta_{ini} + \int_0^t \dot{\theta} \, dt \end{pmatrix}$$

Capítulo 5

Diseño e implementación

En este capítulo se va a documentar todo el proceso seguido para conseguir que el robot pueda ser controlado de forma remota. Desde la adecuación del robot, hasta el desarrollo del código que implementa la interfaz de comunicación.

5.1. Preparación del Neato XV-Essential

Como el uso que se le va a dar al robot va a ser diferente del cual para el que fue desarrollado originalmente, se le han extraído tanto el motor de succión, como el rodillo frontal de recogida de suciedad. De esta forma, obtenemos más espacio en el interior que se usará para emplazar elementos necesarios en su adecuación.

También se le han realizado dos pequeños orificios por los que poder pasar dos cables. Por un lado, el cable USB que conecta con la Raspberry, que se conecta en la parte posterior y accede, gracias al orificio, directamente al interior del robot sin interferir en el haz laser. Y otro cable que se ha soldado directamente sobre los bornes que reciben la corriente de las baterías, al cual se le ha añadido un circuito regulador de tensión para poder alimentar la Raspberry directamente de las baterías a una tensión de 5V.

5.1.1. Circuito regulador de tensión

Surge la necesidad de alimentar la Raspberry ya que es el elemento que actúa cómo interfaz hardware de comunicación. En el inicio se usó una batería externa para móviles para mantener la Raspberry

encendida, pero no era una solución cómoda ya que se debía controlar la carga, tanto de las baterías del robot Neato, como de la batería de móvil que alimentaba la Raspberry para que funcionara correctamente todo el entorno.

Así que se decidió incorporar un circuito regulador de tensión a 5V para poder alimentar la Raspberry directamente de las baterías del Neato, y de esta forma, tener que controlar únicamente el estado de las baterías del Neato. Para ello se montó un circuito regulador de tensión usando un regulador de tensión a 5V. Específicamente se ha usado un regulador STMicroelectronics L7805CV. El circuito se soldó siguiendo los esquemas del fabricante para un uso estándar.

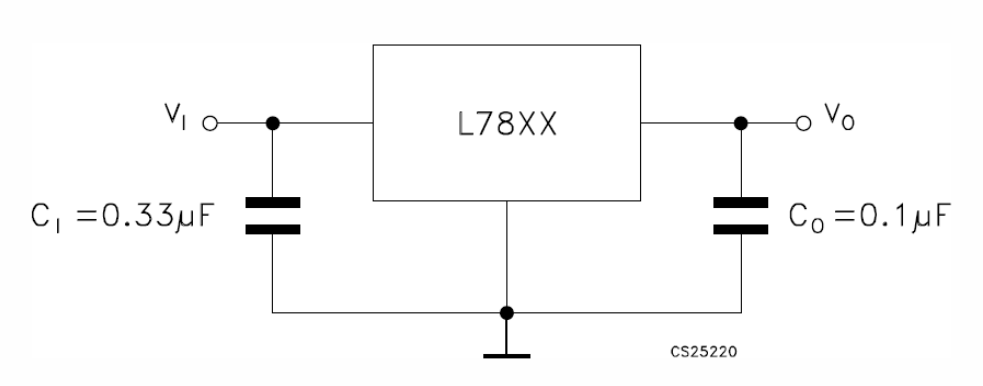

Figura 5.1: Esquema regulador de tensión según STMicroelectronics.

Al cabo de unos días de usar ésta configuración del circuito rectificador, nos dimos cuenta que llegaba un momento en que se perdía la alimentación de la Raspberry. Después de investigar un poco, nos dimos cuenta que el circuito rectificador alcanzaba temperaturas muy elevadas debido a la disipación de potencia provocada por la rectificación del voltaje de las baterías de 14V a la salida de 5V. Por lo que el propio circuito se desconectaba para protegerse del aumento de temperatura.

Por suerte, el propio fabricante, propone en la documentación del circuito un esquema de rectificación con disipación previa de potencia para evitarle tanta carga al componente (5.2). Ésta es la solución que se ha acabado implementando y que funciona correctamente.

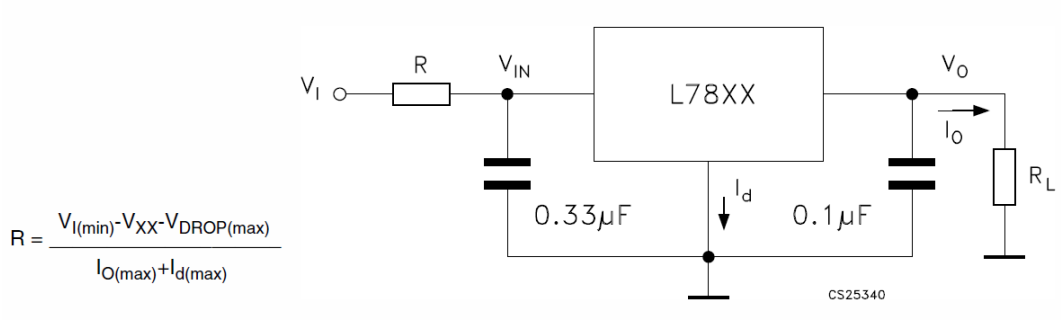

Figura 5.2: Esquema regulador de tensión con filtrado de potencia de entrada según STMicroelectronics.

La resistencia usada en esta nueva configuración, es una resistencia de 12 Ω capaz de disipar 2 Vatios de potencia.

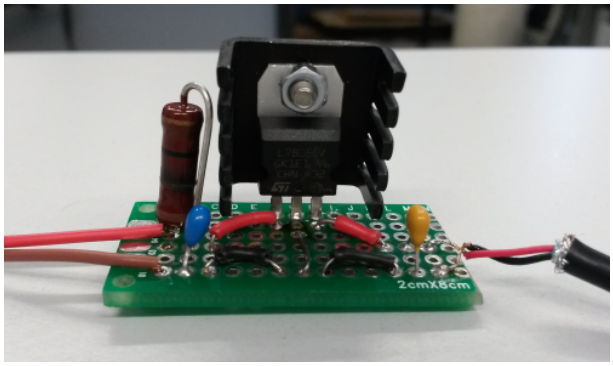

Figura 5.3: Cicuito impreso conteniendo el circuito rectificador de tensión.

5.2. Raspberry Pi 2 Model B

En esta sección se van a detallar todas las configuraciones que se han realizado sobre la Raspberry para poder usarla cómo interfaz de comunicación. Hay que recordar que se le ha añadido una interfaz de comunicación inalámbrica (TP-Link Nano USB Adapter, TL-WN725N), la cual es necesaria configurar, ya que el sistema operativo no la detecta automáticamente. Se explican todas las configuraciones a continuación.

5.2.1. Configuración básica

Para poder usar la Raspberry se ha instalado una distribución estándar de Raspbian y también una versión estándar de Python. No hay problema en usar versiones superiores a las que se indican más adelante, incluso es posible que en versiones superiores de Raspbian no sea necesaria la instalación manual de los drives del adaptador inalámbrico. Las versiones usadas son:

- Raspbian 3.18.7-v7+ #755

- Python 2.7.3

5.2.2. Instalación y configuración de la red inalámbrica

El primer paso para disponer de red inalámbrica es instalar los drivers del adaptador de red. Es necesario conocer qué versión de sistema operativo hay instalado exactamente debido a que los drivers se obtienen de una página web dónde se encuentran todos los de este modelo, pero van empaquetados según la versión del sistema operativo.

Por tanto, el primer paso es averiguar la versión exacta de sistema operativo:

```
uname -a
```

Seguidamente hay que acceder a la página web siguiente y buscar los drivers correspondientes a la versión de sistema operativo:

```
http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=62371
```

Una vez descargada la versión correcta, hay que proceder a su instalación siguiendo los pasos siguientes:

```
1 tar -zxvf 8188eu-20141107.tar.gz #Descomprimir el archivo descargado
2 install -p -m 644 8188eu.ko /lib/modules/$(uname -r)/kernel/drivers/net/
     wireless
3 sudo insmod /lib/modules/$(uname -r)/kernel/drivers/net/wireless/8188eu.ko
4 sudo depmod -a
```

En el comando 1, se descomprime el archivo descargado. En el 2, se instalan los archivos que corresponden a los drivers en su ubicación natural. En el 3 y 4, se registran los módulos instalados para que sean usados por el sistema operativo.

Siguiendo los pasos anteriores, los drivers del adaptador inalámbrico ya estarán instalados y será necesario, a continuación, configurar los parámetros de red para que la Raspberry se conecte a las redes inalámbricas que nos interese al arrancarla. Para ello hay que seguir los pasos siguientes:

Modificar el archivo `\etc\network\interfaces` para que tenga este aspecto:

```
1 auto lo
2 iface lo inet loopback
3 iface eth0 inet dhcp
4
5 allow-hotplug wlan0
6 iface wlan0 inet manual
7 wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
8
9 iface onoes inet dhcp
```

En este archivo de configuración de red, se están definiendo los siguientes comportamientos:

Líneas 1-3, se define la interfaz de loopback "lo", y se activa el DHCP en la interfaz Ethernet "eth0".

En la línea 5, Se configura la conexión y desconexión "en caliente" (sin apagar la máquina) de la interfaz de red inalámbrica "wlan0".

En la línea 6, se define la configuración de red en la interfaz inalámbrica "wlan0" en manual.

En la línea 7, se define el archivo que contendrá las configuraciones de las redes inalámbricas a las que se podrá conectar la Raspberry automáticamente.

En la línea 9, se activa el DHCP para la red "onoes". Ésta red se encuentra definida en el archivo de configuraciones indicado en la línea 7.

Realizadas estas modificaciones, es necesario definir las redes a las que la Raspberry se va a poder conectar automáticamente, para ello, modificamos el archivo definido en el punto 7 anterior `/etc/wpa_supplicant/wpa_supplicant.conf`. Su contenido debe ser el siguiente:

```
1 ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
2 update_config=1
3
4 network={
5     ssid="ONO212902"
6     psk="DD9C392C08"
7     id_str="onoes"
8 }
```

En este archivo de configuración, se están definiendo los siguientes comportamientos:
En las líneas 1-2, se indica que el contenido de éste archivo puede ser modificado dinámicamente por un servicio externo. Cosa que no afecta a nuestra configuración.
Lo importante son las siguientes líneas. Cada configuración de red inalámbrica debe ir dentro de un grupo "network", con los parámetros: "ssid" que indica el nombre de la red inalámbrica a la que queremos conectar, "psk" que indica el password de la red y "id_str" que es el nombre con el que podemos configurar el comportamiento de la red en el archivo de configuración `\etc\network\interfaces`.

Una vez realizadas estas configuraciones, la Raspberry conectará a la red o redes que se le hayan configurado una vez se haya inicializado, obteniendo una IP por DHCP o en su defecto, manteniendo una IP estática definida en su configuración.

5.2.3. Código Python (versión Socket)

Este código es la base de la comunicación con el robot. Por un lado actúa como servidor socket para que se puedan conectar las aplicaciones cliente y controlar el robot, por el otro, establece una conexión serie con el robot a través del cable USB, para pasarle los comandos y obtener la información.

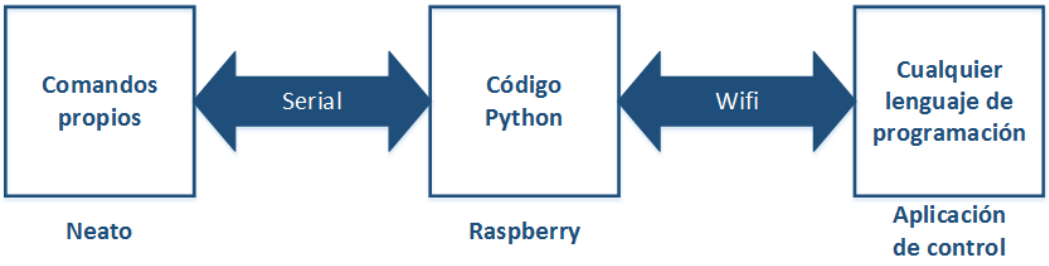

Figura 5.4: Esquema de comunicación Neato - Aplicación.

Debido a que el control se realiza en tiempo real, es muy importante que la comunicación se realice en tiempo real. Por ello, se ha diseñado el código aprovechando los 4 núcleos del procesador de la Raspberry 2 usando técnicas de threading. De esta forma la comunicación es fluida y no se generan cuellos de botella ni esperas en la transmisión de comandos.

El código parte de un hilo de ejecución principal, el cual crea otros 3 hilos de ejecución paralela. Realmente no son threads, sino que son procesos separados, ya que un thread se ejecutaría en el mismo

procesador del hilo que lo crea.

La comunicación entre threads se realiza mediante estructuras de Python "Queue", son unas colas FIFO (First Input First Output) especialmente diseñadas para la comunicación entre threads.

A continuación se detallan los 4 hilos de ejecución que se han implementado:

main_thread: Es el código principal, se encarga de iniciar las estructuras de datos y de crear y destruir el resto de threads cuando es necesario.

serial_thread: Se encarga de la comunicación serie abierta con el robot. Envía los comandos y recibe la respuesta. Le llegan los comandos mediante Queue del slisten_thread. También implementa los modos de funcionamiento explicados más adelante.

slisten_thread: Se encarga de escuchar los comandos enviados por la aplicación cliente mediante el socket y reenviarlos al serial_thread.

swrite_thread: Se encarga de enviar las respuestas a la aplicación cliente mediante el socket.

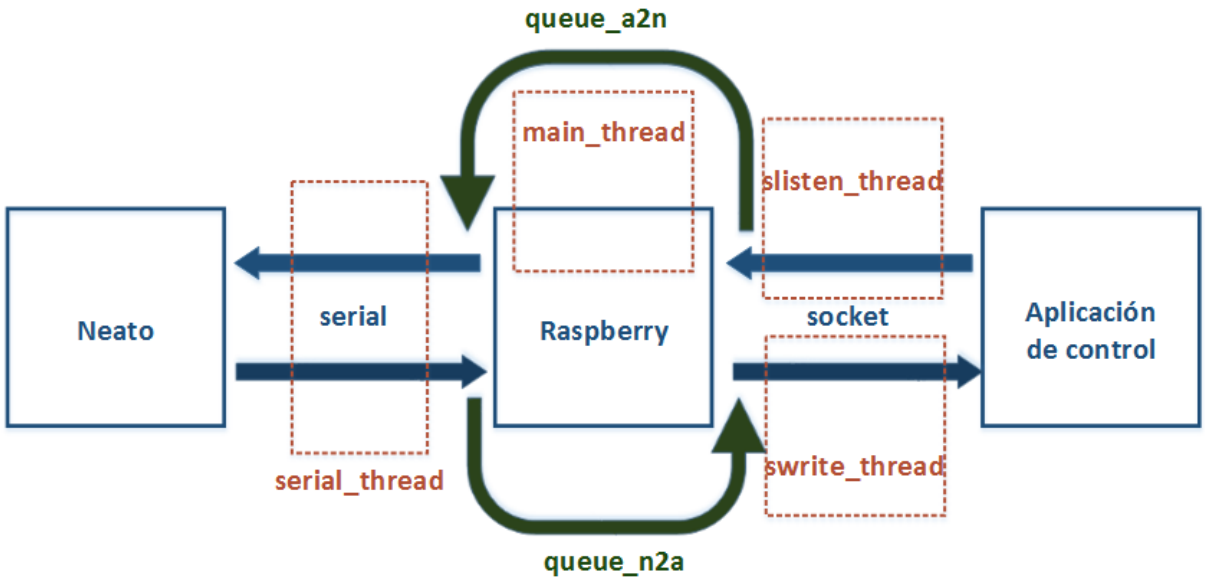

Figura 5.5: Estructura de threads y comunicación.

5.2.4. Código Python (versión WebSocket)

Es idéntico a la versión Socket, pero implementando el protocolo WebSocket según la normalización RFC 6455. Esta implementación establece un canal de comunicación bidireccional y full-duplex sobre un único socket TCP, está diseñada para ser implementada en navegadores y servidores web.

Para establecer una conexión WebSocket, el cliente manda una petición de negociación WebSocket, y el servidor manda una respuesta de negociación WebSocket, como se puede ver en el siguiente ejemplo:

Petición del navegador al servidor:

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4 @1 46546xW%0l 1 5
Origin: http://example.com

^n:ds[4U
```

Respuesta del servidor:

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Location: ws://example.com/demo
Sec-WebSocket-Protocol: sample

8jKS'y:G*Co,Wxa-
```

Los 8 bytes con valores numéricos que acompañan a los campos Sec-WebSocket-Key1 y Sec-WebSocket-Key2 son tokens aleatorios que el servidor utilizará para construir un token de 16 bytes al final de la negociación para confirmar que ha leído correctamente la petición de negociación del cliente.

La negociación o handshake se construye concatenando los números que acompañan al primer campo, y dividiéndolo por el número de espacios en blanco en el propio campo. Esto mismo se repite para el segundo campo. Los dos números resultantes se concatenan entre sí y con los 8 bytes que van después de los campos. El resultado final es una suma MD5 de la cadena concatenada.

Esta negociación puede parecerse a la negociación HTTP, pero no es así. Permite al servidor interpretar parte de la petición de negociación como HTTP y entonces cambiar a WebSocket.

Una vez establecida, las tramas WebSocket de datos pueden empezar a enviarse en ambos sentidos entre el cliente y el servidor en modo full-duplex. Las tramas de texto pueden ser enviadas en modo full-duplex también, en ambas direcciones al mismo tiempo. La información se segmenta en tramas de únicamente 2 bytes. Cada trama empieza con un byte 0x00, termina con un byte 0xFF, y contiene datos UTF-8 entre ellos. Las tramas WebSocket de texto utilizan un terminador, mientras que las tramas binarias utilizan un prefijo de longitud.

5.2.5. Modos de funcionamiento

Una vez establecida la conexión con el socket, se debe definir el modo de funcionamiento mediante la instrucción `DataMode [listener, constant, moody]`. Estos modos de funcionamiento definen el modo en el que interactúan el robot y la aplicación. Se detallan a continuación:

Listener: Modo normal de operación, el robot sólo responde a los comandos enviados desde la aplicación cliente.

Constant: En este modo se envía información de forma continuada sobre todos los sensores del robot sin que tengan que ser solicitados por el cliente. El cliente a su vez puede enviar comandos de control en cualquier momento.

Moody: Es igual que el constant, con la diferencia que se envía un paquete con toda la información y otros dos con sólo la información de las ruedas de forma contínua. La razón de ser de este modo, es la rapidez en obtener los datos. Es mucho más rápido obtener sólo la información de las ruedas que toda la información, por lo que para proyectos dónde la odometría sea fundamental, interesará disponer de más lecturas de ruedas por segundo.

5.2.6. Ejecución en el arranque del código Python

Para no tener que interactuar con la Raspberry, es importante que el servicio de comunicación esté activo una vez se arranca. Por lo que se ha creado una directiva para arrancar el código Python una vez se ha inicializado la Raspberry. Se detalla a continuación el procedimiento para el código Python versión Socket, pero es idéntico para la versión WebSocket.

El primer paso es añadir una línea que invocará la ejecución del código en el inicio, para ello modificamos la configuración del `cron` mediante el archivo `Crontab` en modo superusuario:

```
su crontab -e
```

Dentro de este archivo hay que añadir la siguiente línea:

```
@reboot sh /home/neatoWSRV/nlauncher.sh > /home/neatoWSRV/log.txt
```

Esta línea está indicando lo siguiente: En el arranque, se va a ejecutar el archivo de código sh `nlauncher.sh` y se va a generar un archivo de log `log.txt` en el mismo directorio, conteniendo toda la salida de este archivo sh.

El contenido del archivo `nlauncher.sh` es el siguiente:

```
1 #!/bin/sh
2
3 cd /home/neatoSRV
4 sudo python neatoSRV.py
```

En este archivo, la primera línea indica que el contenido se ejecutará mediante código shell `sh`. La línea 3, realiza un cambio de directorio dónde se encuentra el código Python a ejecutar, y la línea 4, ejecuta en modo superusuario el código Python correspondiente a la interfaz de comunicación.

Con esta serie de configuraciones nos aseguramos de que el servicio de comunicación implementado en el código Python está disponible desde el arranque de la Raspberry.

5.3. Aplicación cliente Matlab

La aplicación Matlab tiene como objetivo servir como herramienta docente en la asignatura de Robótica, tanto en los estudios de Grado como de Master.

Esta herramienta permite controlar el robot en modo joystick y realizar la adquisición de datos. Posteriormente, permite trabajar con los datos ya adquiridos. A continuación se describe el diseño de la interfaz y la interacción con la misma.

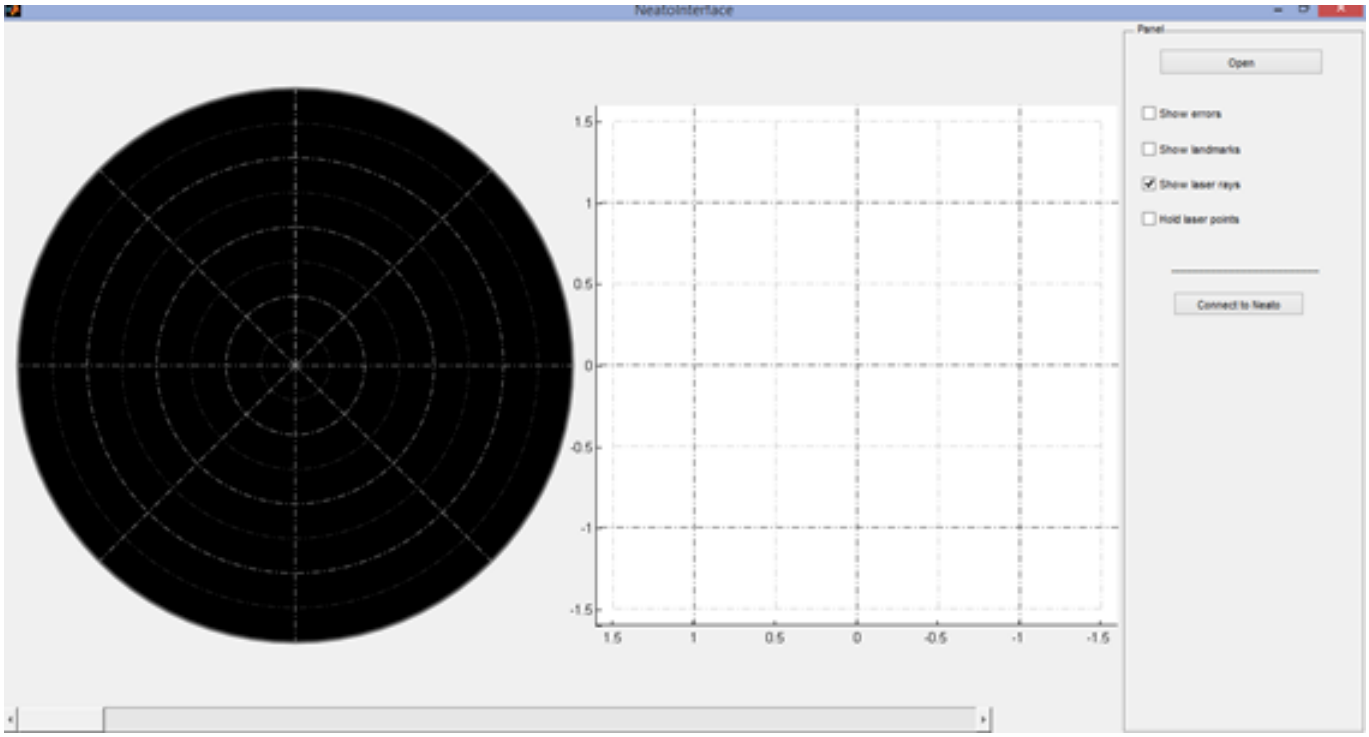

Figura 5.6: Detalle de la interfaz implementada.

En la interfaz se pueden apreciar 2 gráficas principales. La de la izquierda muestra las lecturas del láser respecto el centro del láser, y la de la derecha muestra el entorno en coordenadas mundo. Ésta última, muestra tanto la posición del robot, como la trayectoria y los puntos detectados por el láser. Hay también una barra de desplazamiento lateral en la parte inferior que sirve para avanzar o retroceder sobre las lecturas en modo de visualización.

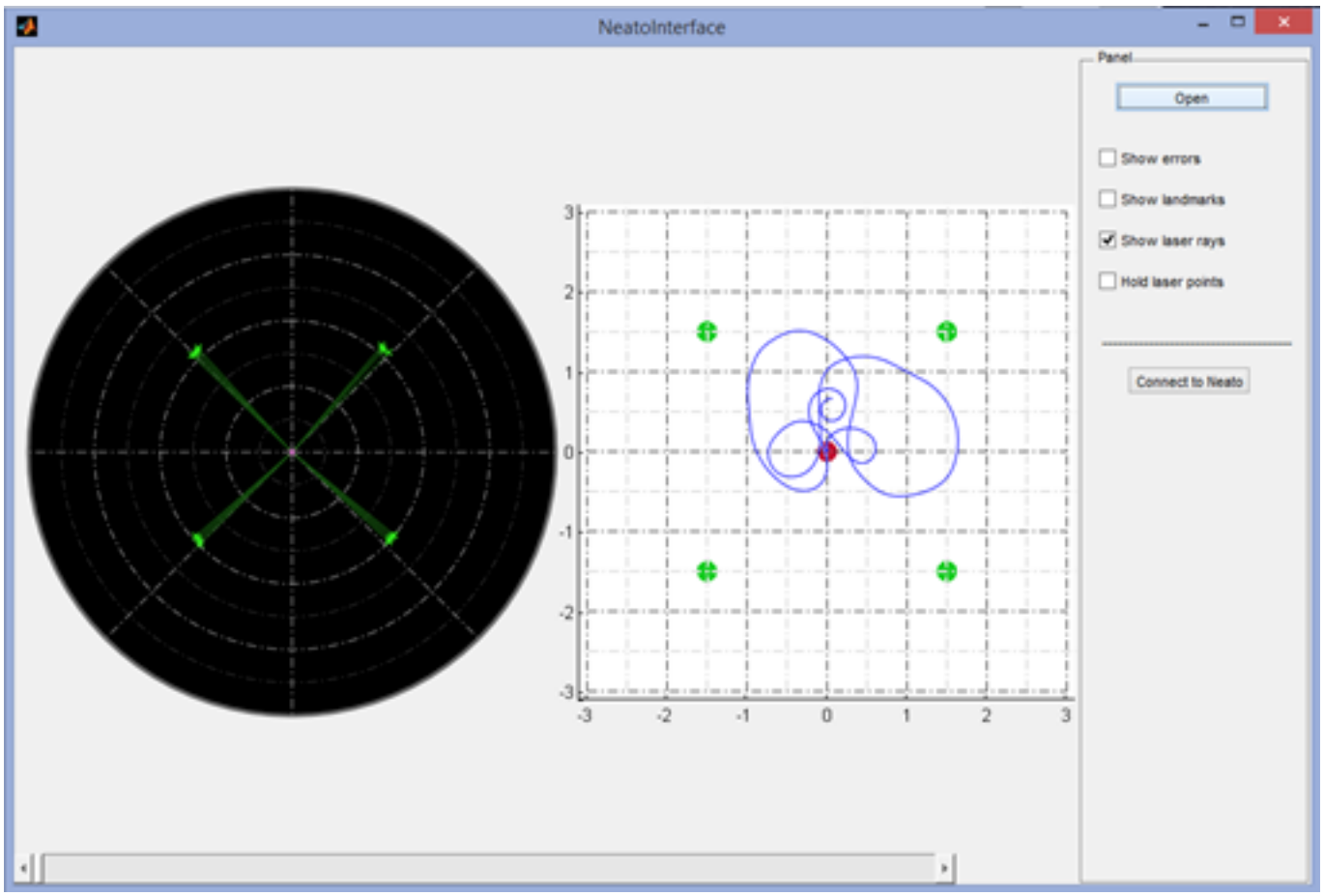

Figura 5.7: Detalle de la interfaz con datos reales.

A continuación se detallan las diferentes funciones que ofrece la interfaz:

5.3.1. Adquisición de datos (Modo joystick)

El proceso de adquisición de datos es muy sencillo, el primer paso es conectar con el robot indicando la IP y el puerto en el que se está publicando el socket de conexión.

Una vez conectado, el láser empezará a girar, se mostrarán los datos en las gráficas de la interfaz y el robot se podrá comandar mediante las teclas de dirección. Las teclas de arriba/abajo, aumentan o disminuyen la velocidad (es posible ir marcha atrás), y las teclas derecha/izquierda, aumentan el ángulo de giro del robot hacia el lado apretado.

Con ello, se puede mover el robot alrededor de un entorno mientras se obtienen los datos de las ruedas y del entorno mediante el láser. Una vez finalizada la adquisición de datos, se debe apretar el botón “Stop”. Todos los datos estarán disponibles en el “workspace” de Matlab, y en unos archivos de

log generados con nombre la fecha actual, que podrá ser usado mediante la interfaz para visualizar la información en momentos posteriores.

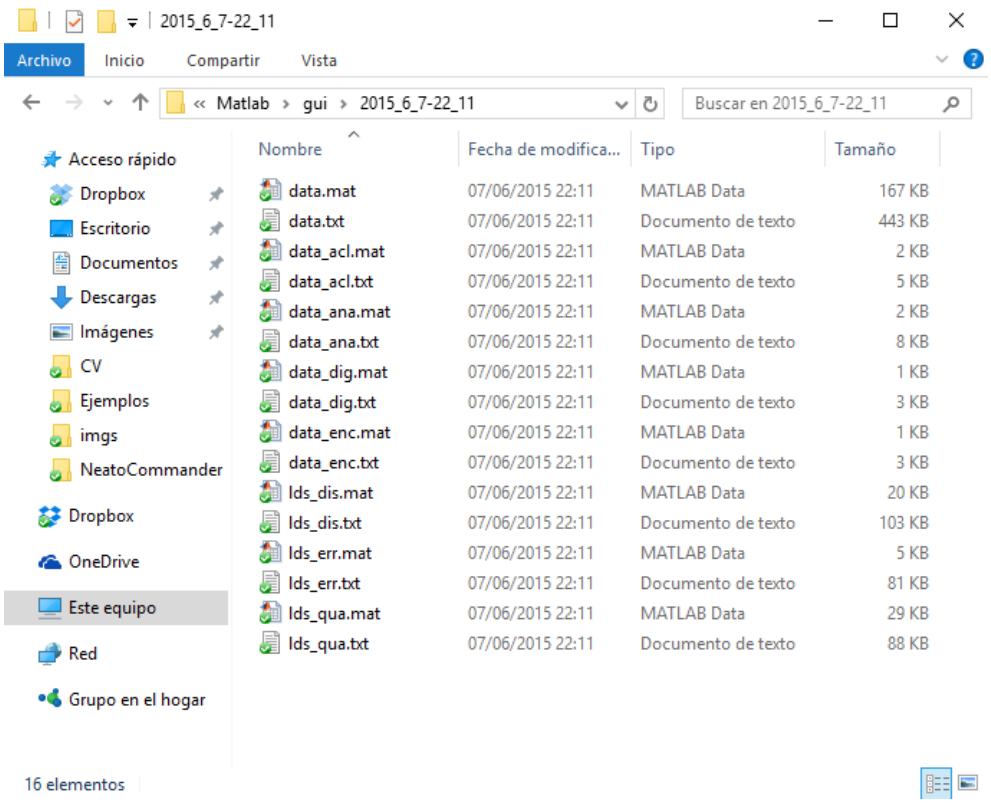

Figura 5.8: Detalle de los archivos generados en la captura de datos.

5.3.2. Visualización y trabajo con los datos

Para poder visualizar y trabajar con datos ya obtenidos, simplemente hay que clicar el botón "Open" de la interfaz, e indicar la carpeta contenedora de los logs con los que se quiera trabajar.

Una vez abierto, los datos estarán disponibles en el "workspace" de Matlab, se podrán visualizar en las gráficas y usar la barra de desplazamiento para navegar por ellos y reproducir el proceso de captura de datos.

5.4. Aplicación cliente Simulink

El objetivo de la aplicación Simulink es disponer de un entorno para crear modelos de control sobre el robot Neato.

Se ha creado una "caja" de Simulink con la que poder controlar el robot. Se ha realizado siguiendo como ejemplo un modelo ya creado existente en la "Robotics Toolbox" de Peter Corke para Simulink. Nos hemos basado en el modelo existente llamado "sl_drivepoint" que utiliza una modelización de una bicicleta para realizar el control sobre ésta.

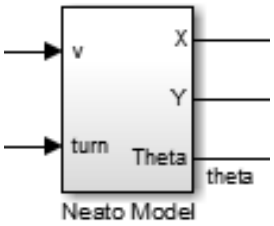

Figura 5.9: Caja de simulink que modeliza el robot e implementa la comunicación.

Esta caja recibe los inputs de velocidad y giro, y retorna los outputs de posición y orientación instantáneas (X, Y y Theta respectivamente). Estos datos serán computados a cada iteración de la simulación que se implemente.

El modelado del robot parte de un código C++ que levanta la conexión socket al inicio de la simulación y que se usará para enviar y recibir datos al robot. A parte de levantar el socket el código C++ (anexo E) realiza las siguientes acciones, y por este orden, en cada iteración:

- Genera la instrucción de movimiento a partir de los parámetros de velocidad y giro de la entrada y la envía al robot.

- Se realiza una lectura del láser y se parsean los datos recibidos.

- Se leen los datos de las ruedas para obtener el desplazamiento y se parsean los datos.

Se realizan estos pasos en este orden, debido a que tiene que pasar un tiempo, desde que se le envía el comando de movimiento, hasta que se lee el estado de las ruedas para que exista un desplazamiento y hayan datos suficientes para poder calcular la nueva posición.

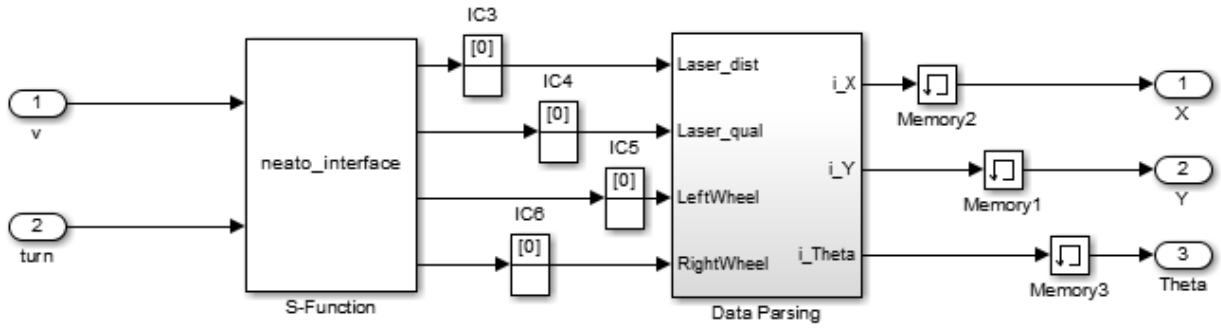

Figura 5.10: Detalle del interior de la caja modelada.

En la imagen anterior se aprecia la implementación de la caja principal. La caja llamada "neato_interface" ejecuta el código C++ comentado anteriormente. Su salida son las lecturas del laser, tanto distancias cómo calidades, y las lecturas de las ruedas. Estos datos se usan en la siguiente caja para calcular la posición del robot. Se muestra la implementación de la caja "Data a Parsing" a continuación.

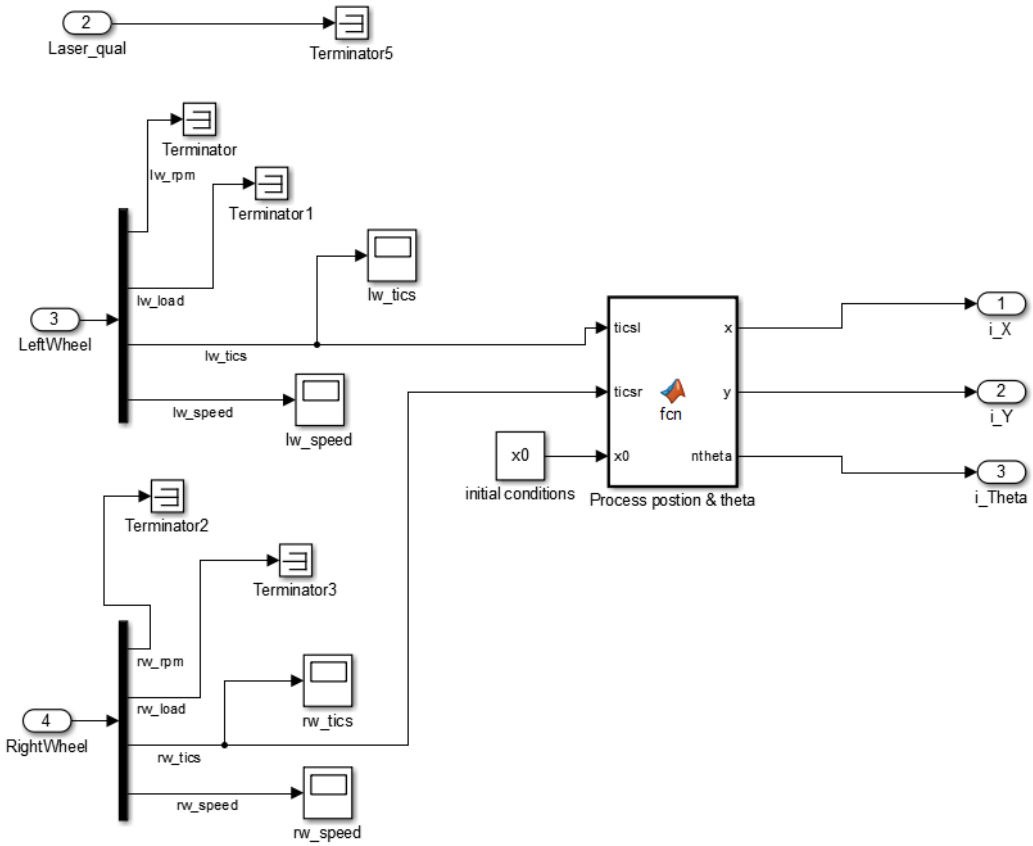

Figura 5.11: Detalle del interior de la caja "Data Parsing".

En la imagen anterior se aprecia cómo los datos de las ruedas van a parar a una caja llamada "Process position & theta". Esta caja está implementada en código M. Y se encarga de mantener la posición y orientación del robot actualizadas en cada momento, a partir de los datos de las ruedas obtenidos. Su salida es la que indica al modelo, la posición y orientación actual del robot.

5.5. Aplicación cliente web

La aplicación cliente web, tiene cómo objetivo ofrecer un entorno de control y de visualizado de datos independientemente de cualquier plataforma de desarrollo.

Las funcionalidades implementadas son las mismas que la interfaz en Matlab, control, obtención de datos y visualización. También se ha implementado un código que realiza la detección de *Landmarks* en tiempo real.

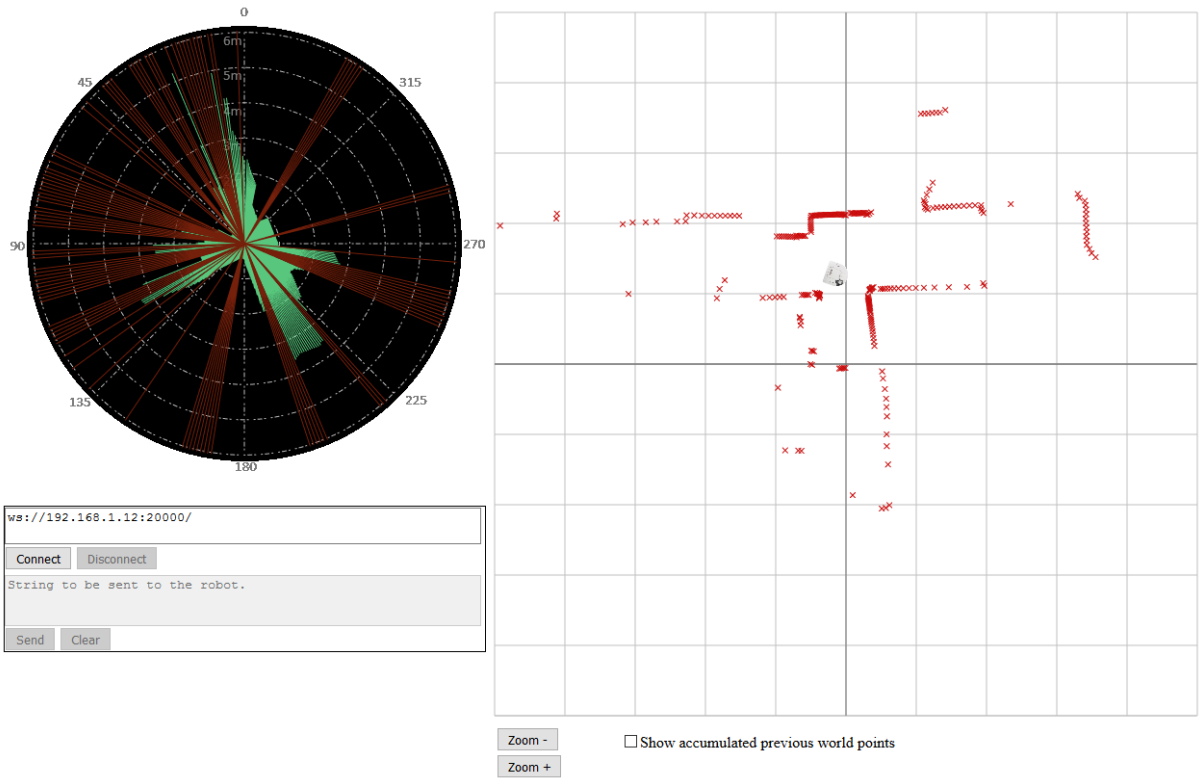

Figura 5.12: Interfaz web.

5.5.1. Web Workers

Se ha usado también una nueva opción de los navegadores, los *Web Workers*, para que la funcionalidad de joystick sea realmente funcional.

Los *Web Workers*, es una solución para ejecutar código fuera del hilo principal de ejecución del navegador. Una clase sencilla de threading.

Debido a la naturaleza de las instrucciones de movimiento que acepta el robot, indicando distancia por rueda y velocidad, éste al realizar el movimiento y alcanzar la distancia indicada, se detiene. Por tanto, para la funcionalidad de joystick no es muy útil, ya que interesaría que mantuviera el movimiento hasta nueva orden.

Se han usado los *Web Workers* para que ese movimiento no se detenga. El Web Worker calcula un 85 % del tiempo que tardará el robot en realizar el último movimiento, y en ese instante, vuelve a enviar el comando para simular un movimiento continuo. En el caso de que se envíe un comando diferente, este reemplaza al anterior en el *Web Worker*, y vuelve a calcular el tiempo.

De esta forma la funcionalidad de joystick es completa. Los *Web Workers* sólo funcionan con el navegador Firefox.

Capítulo 6

Gestión del proyecto

En este capítulo se detalla la planificación y una estimación del coste económico del proyecto.

6.1. Planificación

La siguiente tabla muestra la distribución de las tareas realizadas y el tiempo empleado para llevarlas a cabo.

Tarea	**Tiempo [h]**
Definición del proyecto	**10**
Estudio e investigación	**65**
Estudio del robot Neato	20
Estudio del robot diferencial	20
Estudio de técnica de threading	25
Análisis y diseño	**130**
Definir la interacción con el robot	30
Diseñar el algoritmo de comunicación	50
Diseñar las interficies	50
Implementación	**325**
Modificación del robot	35
Implementar código Python	110
Implementar interfaz Matlab	60
Implementar interfaz Simulink	80
Implementar interfaz web	40
Pruebas	**170**
Documentación	**50**
Tiempo Total	**750**

Cuadro 6.1: Planificación

6.2. Coste económico

En este apartado se realiza un estudio del coste económico de la realización del proyecto como si se hubiera realizado en una empresa. Para realizar este estudio, se va a tener en cuenta el hardware y el software usados y las horas humanas dedicadas según el perfil que debería desempeñar cada tarea.

6.2.1. Hardware

Los costes generados por el hardware, en principio, son fácilmente estimables. Simplemente hay que tener en cuenta el coste de cada elemento físico que se ha usado para realizar el proyecto, y en la mayoría de casos, existe la factura o se puede averiguar su valor.

Componente	**Cantidad**	**Coste**
Portátil	1	1.000,00 €
Neato XV Essential	4	1.260,00 €
Raspberry Pi 2	4	160,00 €
Material diverso	1	150,00 €
Coste Total		**2.570,00 €**

Cuadro 6.2: Coste hardware

6.2.2. Software

El precio de las licencias de software son las que se contabilizan en esta sección. El precio de la licencia de Windows va incluido en el precio del portátil. La licencia de Matlab ha sido una versión de estudiante, que cuesta unos 69 €. Sin embargo, si el proyecto se hubiera implementado en una empresa se tendría que contabilizar el coste de una licencia estándar.

Componente	**Cantidad**	**Coste**
Licencia Matlab	1	2.000,00 €
Coste Total		**2.000,00 €**

Cuadro 6.3: Coste software

6.2.3. Recursos Humanos

En el proyecto han intervenido principalmente el tutor del proyecto y el alumno. El rol del tutor correspondería al de director de proyecto, mientras que el del alumno correspondería al resto de roles

existentes en un proyecto real: analista, programador y tester.

Para realizar el estudio del coste en recursos humanos, se asignan las horas según su tipo a un rol determinado. Las horas de definición del proyecto se han asignado al rol de director; las de estudio e investigación, y análisis y diseño se han asignado al rol de analista; las horas de implementación al rol de programador y las pruebas, al rol de tester. Las horas dedicadas a la documentación se han repartido entre todos los roles, ya que cada uno debe documentar parte de su trabajo.

Rol	**Horas**	**Precio/Hora**	**Coste**
Director	20	70,00 €/hora	1.400,00 €
Analista	215	50,00 €/hora	10.750,00 €
Programador	330	30,00 €/hora	9.900,00 €
Tester	175	20,00 €/hora	3.500,00 €
Coste Total			**25.550,00 €**

Cuadro 6.4: Coste recursos humanos

6.2.4. Coste total

Para conocer el coste total del proyecto debemos agrupar todos los costes anteriores, obteniendo un coste total de: 30.120,00 €.

Tipo de coste	**Coste**
Hardware	2.570,00 €
Software	2.000,00 €
Recursos humanos	25.550,00 €
Coste Total	**30.120,00 €**

Cuadro 6.5: Coste total

Capítulo 7

Conclusiones

En este capítulo se detallan los objetivos cumplidos y posibles trabajos futuros o mejoras.

7.1. Objetivos alcanzados

Se detallan a continuación los objetivos que han sido resueltos con éxito durante la elaboración de este proyecto.

7.1.1. Preparación del robot

Las modificaciones realizadas sobre el robot, la incorporación de la Raspberry Pi 2 y el código de comunicación Python, han funcionado correctamente convirtiendo un robot de limpieza comercial en una plataforma de desarrollo para proyectos de robótica móvil o para su uso en docencia.

7.1.2. Implementación de la interfaz Matlab

La interfaz Matlab permite el control del robot obteniendo datos del entorno y del movimiento del robot, también el uso posterior de los datos para su visualización y tratamiento, convirtiéndola en una herramienta muy útil para la docencia en robótica.

7.1.3. Implementación del entorno Simulink

El modelo del robot Neato implementado para Simulink permite integrar el robot como si fuera un elemento propio de este entorno, de forma que se pueden crear una infinita variedad de modelos de control exportables a otras plataformas.

7.1.4. Implementación de la interfaz web

La interfaz web implementa la conexión WebSocket, permitiendo controlar el robot y visualizar la información desde un navegador web. De esta forma se obtiene una interfaz de trabajo que no requiere de un entorno software como Matlab.

7.2. Trabajos fututos

En este apartado hay que diferenciar entre el robot, las interfaces con finalidad docente, la interfaz web y Simulink. Sobre el robot, se le podría añadir algún otro tipo de sensor o usar algunos sensores que no se han usado en este proyecto, como son el acelerómetro, los sensores anti-caída y los bumpers.

Sobre las aplicaciones con finalidad docente, se pueden implementar más funcionalidades, cómo un simulador del robot para poder explorar diferentes entornos sin disponer del robot físico. U otro simulador que responda a una aplicación de control externa, es decir, que en lugar de conectarse al robot, se conectara al simulador del robot para ver cómo actuaría.

Sobre simulink, las posibilidades son infinitas. Per se es un entorno de creación de modelos de control, así que se pueden desarrollar multitud de proyectos de control con este entorno.

Sobre la interfaz web, sería interesante implementar también un simulador del robot, al igual que una plataforma de programación del robot, para disponer de un entorno de entrenamiento o testeo de programación de inteligencia artificial sobre robots móviles.

Anexo A

Manual del programador para Neato.

Programmer's Manual

Communication with the robot through the USB port
The robot has a USB port next to the power jack at the rear of the robot. You will need a USB 5-pin mini cable to connect to the robot. You will also need to install a device driver to talk with the robot. The best way to install the device driver is to install it from the **Software Update Page**.
Communication to a robot through a terminal emulator
Commands can be sent to a real robot via a terminal emulator. The robot is running an API command interpreter on the USB port. Connection to the robot is made through a USB cable. The driver converts the USB port connection to a Com port connection. This allows you to use any terminal emulator program (such as Hyperterm or TeraTerm) to communicate with the robot. Use the following procedure:

1. Turn on the robot and connect the USB 5 pin mini cable between your computer and your robot.

2. Bring up the Terminal Emulator program.

3. Find the COM port that the robot is connected to. This is usually the last one on the list. The communication parameters (Baud, start/stop bits, parity, etc.) are unimportant. They apply only to a real COM port.

4. Type GetVersion. If you are connected to the robot, then it will echo back each character that you type. Hit the enter key. The robot will now process the command and supply a response.

5. Now type help to get a list of available commands.

Command Syntax

Commands are matched with a case insensitive method. Partial word match is supported. You only need to type in enough letters of the command to make it unique. Commands have a flexible format. In the strictly non-sequenced format, the syntax consists of only 3 elements:

Cmd – This is the command name.
Flag – Flags are always initialized to zero. Specifying a flag on the command line sets the flag to one.
ParamName ParamValue – The ParamName specifies what parameter the next argument (ParamValue) is.

Flags and Param(Name/Value) pairs can be specified on the command line in any order. All Flags and ParamNames are matched with a case insensitve method. Partial word match is supported. You only need to type in enough letters of the Flag or ParamName to make it unique. In a less verbose format, the user can omit the ParamName from the Param(Name/Value) pairs. However, this format requires that all ParamValues are specified in the correct sequence. This sequence is the sequence shown in the Usage command. Unnamed ParamValues are assigned in sequence to the earliest ParamName in the sequence that does not already have a value assigned.

Response Syntax

All responses will have a control-Z (^Z) at the end of the response string. Command responses are in the form of CSV spread sheets. E.G. GetVersion returns:

Component,Major,Minor,Build
Product Model,XV-11,,
Serial Number,AAAnnnnnAA,0000000,D
Software,6,1,13328
LDS Software,V1.0.0,,
LDS Serial,XXX-YYY,,
MainBoard Vendor ID,1,,

MainBoard Serial Number,99,,
MainBoard Version,0,8,
Chassis Version,-1,,
UIPanel Version,-1,,

The first line of text is the column labels. Each line afterwards is composed of a row label, a comma, and then data associated with that label. The order and number of the rows and columns is not gauranteed to stay the same from release to release. The labels used will not be changed, but new rows and columns may be added anywhere. The application should parse the full response into rows and columns, and look up the particular value by matching its row and column names in the table.

Table of Robot Application Commands

Command	Description
Clean	Starts a cleaning by simulating press of start button.
DiagTest	Executes different test modes. Once set, press Start button to engage. (Test modes are mutually exclusive.)
GetAccel	Get the Accelerometer readings.
GetAnalogSensors	Get the A2D readings for the analog sensors.
GetButtons	Get the state of the UI Buttons.
GetCalInfo	Prints out the cal info from the System Control Block.
GetCharger	Get the diagnostic data for the charging system.
GetDigitalSensors	Get the state of the digital sensors.
GetErr	Get Error Message.
GetLDSScan	Get scan packet from LDS.
GetLifeStatLog	Get All Life Stat Logs.
GetMotors	Get the diagnostic data for the motors.
GetSchedule	Get the Cleaning Schedule. (24 hour clock format)
GetSysLog	Get System Log data.
GetTime	Get Current Scheduler Time.
GetVersion	Get the version information for the system software and hardware.
GetWarranty	Get the warranty validation codes.
Help	Without any argument, this prints a list of all possible cmds. With a command name, it prints the help for that particular command
PlaySound	Play the specified sound in the robot.
RestoreDefaults	Restore user settings to default.
SetDistanceCal	Set distance sensor calibration values for min and max distances.
SetFuelGauge	Set Fuel Gauge Level.
SetLCD	Sets the LCD to the specified display. (TestMode Only)
SetLDSRotation	Sets LDS rotation on or off. Can only be run in TestMode.
SetLED	Sets the specified LED to on,off,blink, or dim. (TestMode Only)
SetMotor	Sets the specified motor to run in a direction at a requested speed. (TestMode Only)
SetSchedule	Modify Cleaning Schedule.
SetSystemMode	Set the operation mode of the robot. (TestMode Only)
SetTime	Sets the current day, hour, and minute for the scheduler clock.
SetWallFollower	Enables/Disables wall follower
TestMode	Sets TestMode on or off. Some commands can only be run in TestMode.
Upload	Uploads new program to the robot.

Detailed Command Descriptions

Command: Clean

Description: Starts a cleaning by simulating press of start button.
Usage: Clean [House] [Spot] [Stop]
Options:

Flag	Description
House	(Optional) Equivalent to pressing 'Start' button once. Starts a house cleaning. (House cleaning mode is the default cleaning mode.)
Spot	(Optional) Starts a spot clean.
Stop	Stop Cleaning.

Return Format: None

Command: DiagTest

Description: Executes different test modes. Once set, press Start button to engage. (Test modes are mutually exclusive.)
Usage: DiagTest [TestsOff] [DrivePath] [DriveForever] [MoveAndBump] [DropTest] [AutoCycle] [OneShot] [BrushOn] [VacuumOn] [LDSOn] [AllMotorsOn] [DisablePickupDetect] [DrivePathDist <DrivePathDist_value>] [DriveForeverLeftDist <DriveForeverLeftDist_value>] [DriveForeverRightDist <DriveForeverRightDist_value>] [DriveForeverSpeed <DriveForeverSpeed_value>] [Speed <Speed_value>] [BrushSpeed <BrushSpeed_value>]
Options:

Flag	Description
TestsOff	Stop Diagnostic Test and clears all diagnostic test modes.
DrivePath	Sets DrivePath TestMode. Press start button to start. Robot travels straight by commanded distance as path. Mutually exclusive with other diagtest modes. Use 'TestsOff' option to stop.
DriveForever	Sets DriveForever TestMode. Press start button to start. Robot drives continuously. Mutually exclusive with other diagtest modes. Use 'TestsOff' option to stop.
MoveAndBump	Sets Move and Bump TestMode. Press start button to start. Executes canned series of motions, but will react to bumps. Mutually exclusive with other diagtest modes.
DropTest	Enables DropTest. Robot drives forward until a drop is detected. Mutually exclusive with other diagtest modes.
AutoCycle	DropTest argument to enable automatic restart of the test. The robot will drive backwards and then forward until a drop is detected until the test is over.
OneShot	Only executes test once.
BrushOn	Turns on brush during test. May conflict with motor commands of test so use carefully!
VacuumOn	Turns on vacuum during test. May conflict with motor commands of test so use carefully!

LDSOn	Turns on LDS during test. May conflict with motor commands of test so use carefully!
AllMotorsOn	Turns on brush, vacuum, and lds during test. May conflict with motor commands of test so use carefully!
DisablePickupDetect	Ignores pickup (wheel suspension). By default, pickup detect is enabled and stops the test.
DrivePathDist	Distance in mm
DriveForeverLeftDist	Use next arg to set left wheel dist for DriveForever test. Requires DriveForeverRightDist as well. The ratio of this value to DriveForeverRightDist determines turn radius.
DriveForeverRightDist	Use next arg to set right wheel dist for DriveForever test. Requires DriveForeverLeftDist as well. The ratio of this value to DriveForeverLeftDist determines turn radius.
DriveForeverSpeed	Use next arg to set turn speed of outer wheel for DriveForever test in mm/s.
Speed	DropTest argument to set the robot speed in mm/s.
BrushSpeed	DropTest argument to set the speed of the brush in rpm.

Return Format: None

Command: GetAccel

Description: Get the Accelerometer readings.
Usage: GetAccel
Options: None
Return Format:

```
Label,Value PitchInDegrees,  0.00 RollInDegrees,  0.00 XInG, 0.000 YInG, 0.000 ZInG,
0.000 SumInG, 0.000
```

Command: GetAnalogSensors

Description: Get the A2D readings for the analog sensors.
Usage: GetAnalogSensors [raw] [stats]
Options:

Flag	Description
raw	Return raw analog sensor values as milliVolts. (Default is sensor values in native units of what they measure.)
stats	Return stats (avg,max,min,dev,cnt) of raw analog sensor values as milliVolts. (Implies 'raw' option)

Return Format:

```
'GetAnalogSensors' produces:  SensorName,Value WallSensorInMM,34585
BatteryVoltageInmV,19761 LeftDropInMM,60 RightDropInMM,60 RightMagSensor,0
LeftMagSensor,0 XTemp0InC,28 XTemp1InC,28 VacuumCurrentInmA,1342 ChargeVoltInmV,0
NotConnected1,0 BatteryTemp1InC,20 NotConnected2,0 CurrentInmA,3493 NotConnected3,0
```

```
BatteryTemp0InC,20  'GetAnalogSensors raw' produces:  SensorName,SignalVoltageInmV
WallSensorInMM,0 BatteryVoltageInmV,2574 LeftDropInMM,3296 RightDropInMM,3296
RightMagSensor,0 LeftMagSensor,0 XTemp0InC,1759 XTemp1InC,1759 VacuumCurrentInmA,322
ChargeVoltInmV,0 NotConnected1,0 BatteryTemp1InC,1759 NotConnected2,0
CurrentInmA,992 NotConnected3,0 BatteryTemp0InC,1759  'GetAnalogSensors stats'
produces:  SensorName,Mean,Max,Min,Cnt,Dev WallSensorInMM,0,0,0,50,0
BatteryVoltageInmV,2574,2574,2574,50,0 LeftDropInMM,3296,3296,3296,50,0
RightDropInMM,3296,3296,3296,50,0 RightMagSensor,0,0,0,50,0 LeftMagSensor,0,0,0,50,0
XTemp0InC,1759,1759,1759,50,0 XTemp1InC,1759,1759,1759,50,0
VacuumCurrentInmA,322,322,322,50,0 ChargeVoltInmV,0,0,0,50,0
NotConnected1,0,0,0,50,0 BatteryTemp1InC,1759,1759,1759,50,0
NotConnected2,0,0,0,50,0 CurrentInmA,992,992,992,50,0 NotConnected3,0,0,0,50,0
BatteryTemp0InC,1759,1759,1759,50,0
```

Command: GetButtons

Description: Get the state of the UI Buttons.
Usage: GetButtons
Options: None
Return Format:

```
Button Name,Pressed BTN_SOFT_KEY,0 BTN_SCROLL_UP,0 BTN_START,0 BTN_BACK,0
BTN_SCROLL_DOWN,0
```

Command: GetCalInfo

Description: Prints out the cal info from the System Control Block.
Usage: GetCalInfo
Options: None
Return Format:

```
Parameter,Value LDSOffset,0 XAccel,0 YAccel,0 ZAccel,0 RTCOffset,0 LCDContrast,43
RDropMin,-1 RDropMid,-1 RDropMax,-1 LDropMin,-1 LDropMid,-1 LDropMax,-1 WallMin,-1
WallMid,-1 WallMax,-1
```

Command: GetCharger

Description: Get the diagnostic data for the charging system.
Usage: GetCharger
Options:

Flag	Description

Return Format:

```
Charger Variable Name, Value Label,Value FuelPercent,100 BatteryOverTemp,0
ChargingActive,0 ChargingEnabled,0 ConfidentOnFuel,0 OnReservedFuel,0 EmptyFuel,0
BatteryFailure,0 ExtPwrPresent,0 ThermistorPresent[0],0 ThermistorPresent[1],0
BattTempCAvg[0],103 BattTempCAvg[1],103 VBattV,0.21 VExtV,0.00 Charger_mAH,0
MaxPWM,65536 PWM,-858993460
```

Command: GetDigitalSensors

Description: Get the state of the digital sensors.
Usage: GetDigitalSensors
Options: None
Return Format:

```
Digital Sensor Name, Value SNSR_DC_JACK_CONNECT,0 SNSR_DUSTBIN_IS_IN,1
SNSR_LEFT_WHEEL_EXTENDED,0 SNSR_RIGHT_WHEEL_EXTENDED,0 LSIDEBIT,0 LFRONTBIT,0
RSIDEBIT,0 RFRONTBIT,0
```

Command: GetErr

Description: Get Error Message.
Usage: GetErr [Clear]
Options:

Flag	Description
Clear	Dismiss the reported error.

Return Format:

```
If an error currently exists, then report the error message. Otherwise no msg.
Possible Error Msgs are:  1 - WDT 2 - SSEG LED 3 - BTN LED 4 - BACK LED 5 - FLASH 10
- BattNominal 11 - BattOverVolt 12 - BattUnderVolt 13 - BattOverTemp 14 -
BattShutdownTemp 15 - BattUnderCurrent 16 - BattTimeout 17 - BattTempPeak 18 -
BattFastCapacity 19 - BattMACapacity 20 - BattOnReserve 21 - BattEmpty 22 -
BattMismatch 23 - BattLithiumAdapterFailure 207 - I had to reset my system. Please
press START to clean 217 - Please unplug my Power Cable when you want me to clean.
218 - Please unplug my USB Cable when you want me to clean. 219 - Please set
schedule to ON first. 220 - Please set my clock first. 222 - Please put my Dirt Bin
back in. 223 - Please check my Dirt Bin and Filter. Empty them as needed. 224 - My
Brush is overheated. Please wait while I cool down. 225 - My Battery is overheated.
Please wait while I cool down. 226 - I am unable to navigate. Please clear my path.
227 - Please return me to my base. 228 - My Bumper is stuck. Please free it. 229 -
Please put me down on the floor. 230 - I can't charge. Try moving the base station
to a new location 231 - My Left Wheel is stuck. Please free it from debris. 232 - My
Right Wheel is stuck. Please free it from debris. 233 - I have an RPS error. Please
visit web support. 234 - My Brush is stuck. Please free it from debris. 235 - My
Brush is overloaded. Please free it from debris. 236 - My Vacuum is stuck. Please
visit web support. 237 - Please Check my filter and Dirt Bin 238 - My Battery has a
critical error. Please visit web support. 239 - My Brush has a critical error.
Please visit web support. 240 - My Schedule is now OFF 241 - I can't shut down while
I am connected to power. 243 - A Software update is available. Please visit web
support. 244 - My SCB was corrupted. I reinitialized it. Please visit web support.
245 - Please Dust me off so that I can see.
```

Command: GetLDSScan

Description: Get scan packet from LDS.
Usage: GetLDSScan
Options:

Flag	Description

Return Format:

```
360 output lines of LDS Scan Angle, Distance code in MM, normalized spot intensity,
and error code. Followed by 2 status variable pairs. Example:
AngleInDegrees,DistInMM,Intensity,ErrorCodeHEX 0,221,1400,0 1,223,1396,0
2,228,1273,0 (. . .) 359,220,1421,0 ROTATION_SPEED (in Hz, Floating Point),5.00
```

Command: GetLifeStatLog

Description: Get All Life Stat Logs.
Usage: GetLifeStatLog
Options: None
Return Format:

```
Multiple LifeStat logs are output, from the oldest to the newest.   Note that only
the non-zero entries are printed.  runID,statID,count,Min,Max,Sum,SumV*2
0,LS_RunDate,3,-3,0,0x0,0x0 0,LS_A2D0,3,-2,2,0x3,0x3 0,LS_A2D1,3,-1,4,0x6,0xc
0,LS_A2D2,3,0,6,0x9,0x1b 0,LS_A2D3,3,1,8,0xc,0x30 0,LS_A2D4,3,2,10,0xf,0x4b
0,LS_A2D5,3,3,12,0x12,0x6c 0,LS_A2D6,3,4,14,0x15,0x93 0,LS_A2D7,3,5,16,0x18,0xc0
0,LS_A2D8,3,6,18,0x1b,0xf3 0,LS_A2D9,3,7,20,0x1e,0x12c 0,LS_A2D10,3,8,22,0x21,0x16b
0,LS_A2D11,3,9,24,0x24,0x1b0 0,LS_A2D12,3,10,26,0x27,0x1fb
0,LS_A2D13,3,11,28,0x2a,0x24c 0,LS_A2D14,3,12,30,0x2d,0x2a3
0,LS_A2D15,3,13,32,0x30,0x300 0,LS_LDROP_MM,3,14,34,0x33,0x363
0,LS_RDROP_MM,3,15,36,0x36,0x3cc 0,LS_CLEANTYPE,3,16,38,0x39,0x43b
0,LS_ERROR_BRUSH_OVERTEMP,3,17,40,0x3c,0x4b0
0,LS_ERROR_BATTERY_OVERTEMP,3,18,42,0x3f,0x52b
0,LS_ERROR_LWHEEL_STUCK,3,19,44,0x42,0x5ac
0,LS_ERROR_RWHEEL_STUCK,3,20,46,0x45,0x633 0,LS_ERROR_LDS_JAMMED,3,21,48,0x48,0x6c0
0,LS_ERROR_BRUSH_STUCK,3,22,50,0x4b,0x753 0,LS_ERROR_VACUUM_STUCK,3,23,52,0x4e,0x7ec
0,LS_ERROR_BRUSH_SLIP,3,24,54,0x51,0x88b 0,LS_ERROR_VACUUM_SLIP,3,25,56,0x54,0x930
0,LS_ERROR_BATTERY_CRITICAL,3,26,58,0x57,0x9db
0,LS_ERROR_LDROP_STUCK,3,27,60,0x5a,0xa8c 0,LS_ERROR_RDROP_STUCK,3,28,62,0x5d,0xb43
0,LS_ERROR_WARNING,3,29,64,0x60,0xc00 0,LS_ERROR_SHUTDOWN,3,30,66,0x63,0xcc3
0,LS_NEW_BATTERY,3,31,68,0x66,0xd8c 0,LS_ERROR_WDT,3,32,70,0x69,0xe5b
0,LS_LDS_DotTooWide,3,33,72,0x6c,0xf30 0,LS_LDS_DoubleDot,3,34,74,0x6f,0x100b
0,LS_LDS_TooNear,3,35,76,0x72,0x10ec 0,LS_LDS_OutOfCal,3,36,78,0x75,0x11d3
0,LS_LDS_TooFar,3,37,80,0x78,0x12c0 0,LS_LDS_TooDim,3,38,82,0x7b,0x13b3
0,LS_LDS_NoReading,3,39,84,0x7e,0x14ac 0,LS_LDS_TooSlow,3,40,86,0x81,0x15ab
0,LS_LDS_EncoderSkip,3,41,88,0x84,0x16b0 0,LS_LDS_DumbPanavision,3,42,90,0x87,0x17bb
0,LS_LDS_SunBlind,3,43,92,0x8a,0x18cc 0,LS_LDS_NoDot,3,44,94,0x8d,0x19e3
0,LS_LDS_BadArg,3,45,96,0x90,0x1b00 0,LS_LDS_NoMatch,3,46,98,0x93,0x1c23
0,LS_LDS_Abort,3,47,100,0x96,0x1d4c 0,LS_LDS_Escape,3,48,102,0x99,0x1e7b
0,LS_LDS_LaserCharging,3,49,104,0x9c,0x1fb0
0,LS_LDS_LaserPhotoTrip,3,50,106,0x9f,0x20eb
0,LS_LDS_LaserEncoderTrip,3,51,108,0xa2,0x222c
0,LS_LDS_LaserClockTrip,3,52,110,0xa5,0x2373
0,LS_LDS_LaserDebugTrip,3,53,112,0xa8,0x24c0
0,LS_LDS_LaserOverCurrent,3,54,114,0xab,0x2613
0,LS_LDS_LaserOverPWM,3,55,116,0xae,0x276c
0,LS_LDS_LaserPhotoStuck,3,56,118,0xb1,0x28cb
0,LS_LDS_LaserCurrentStuck,3,57,120,0xb4,0x2a30
0,LS_LDS_LaserOverTemp,3,58,122,0xb7,0x2b9b
0,LS_LDS_LaserFlakyPhoto,3,59,124,0xba,0x2d0c
0,LS_LDS_LaserFlakyCurrent,3,60,126,0xbd,0x2e83
0,LS_LDS_ReservedCode,3,61,128,0xc0,0x3000 0,LS_ERROR_DFLT_APP,3,62,130,0xc3,0x3183
0,LS_ERROR_CORRUPT_SCB,3,63,132,0xc6,0x330c
0,LS_ERROR_BRUSH_CRITICAL,3,64,134,0xc9,0x349b
0,LS_ERROR_DECK_DEBRIS,3,65,136,0xcc,0x3630 0,LS_DOCKED,3,66,138,0xcf,0x37cb
0,LS_LMAG,3,67,140,0xd2,0x396c 0,LS_RMAG,3,68,142,0xd5,0x3b13
0,LS_ALERT_RETURN_TO_BASE,3,69,144,0xd8,0x3cc0
0,LS_ALERT_RETURN_TO_BASE_PWR,3,70,146,0xdb,0x3e73
0,LS_ALERT_RETURN_TO_START,3,71,148,0xde,0x402c
0,LS_ALERT_RETURN_TO_CHARGE,3,72,150,0xe1,0x41eb
0,LS_ALERT_DUST_BIN_FULL,3,73,152,0xe4,0x43b0
```

```
0,LS_ALERT_BUSY_CHARGING,3,74,154,0xe7,0x457b
0,LS_ALERT_RECOVERING_LOCATION,3,75,156,0xea,0x474c
0,LS_ALERT_CHARGING_POWER,3,76,158,0xed,0x4923
0,LS_ALERT_CHARGING_BASE,3,77,160,0xf0,0x4b00
0,LS_ALERT_CONNECT_CHRG_CABLE,3,78,162,0xf3,0x4ce3
0,LS_ERROR_DISCONNECT_CHRG_CABLE,3,79,164,0xf6,0x4ecc
0,LS_ERROR_DUST_BIN_EMPTIED,3,80,166,0xf9,0x50bb
0,LS_ERROR_DUST_BIN_MISSING,3,81,168,0xfc,0x52b0
0,LS_ERROR_DUST_BIN_FULL,3,82,170,0xff,0x54ab
0,LS_ERROR_UNABLE_TO_NAVIGATE,3,83,172,0x102,0x56ac
0,LS_ERROR_UNABLE_TO_RETURN_TO_BASE,3,84,174,0x105,0x58b3
0,LS_ERROR_BUMPER_STUCK,3,85,176,0x108,0x5ac0
0,LS_ERROR_PICKED_UP,3,86,178,0x10b,0x5cd3
0,LS_ERROR_RECONNECT_FAILED,3,87,180,0x10e,0x5eec 0,LS_WALL_MM,3,88,182,0x111,0x610b
0,LS_LDROP,3,89,184,0x114,0x6330 0,LS_RDROP,3,90,186,0x117,0x655b
0,LS_LSBUMP,3,91,188,0x11a,0x678c 0,LS_LFBUMP,3,92,190,0x11d,0x69c3
0,LS_RFBUMP,3,93,192,0x120,0x6c00 0,LS_RSBUMP,3,94,194,0x123,0x6e43
0,LS_LDS_Corruption,3,95,196,0x126,0x708c
0,LS_ERROR_DISCONNECT_USB_CABLE,3,96,198,0x129,0x72db
```

Command: GetMotors

Description: Get the diagnostic data for the motors.
Usage: GetMotors [Brush] [Vacuum] [LeftWheel] [RightWheel] [Laser] [Charger]
Options:

Flag	Description
Brush	Return Brush Motor stats.
Vacuum	Return Vacuum Motor stats.
LeftWheel	Return LeftWheel Motor stats.
RightWheel	Return RightWheel Motor stats.
Laser	Return LDS Motor stats.
Charger	Return Battery Charger stats.

Return Format:

```
If no flags are specified, then all motors are reported on. Parameter,Value
Brush_MaxPWM,65536 Brush_PWM,-858993460 Brush_mVolts,1310 Brush_Encoder,0 Brush_RPM,
-858993460 Vacuum_MaxPWM,65536 Vacuum_PWM,-858993460 Vacuum_CurrentInMA,52428
Vacuum_Encoder,0 Vacuum_RPM, 52428 LeftWheel_MaxPWM,65536 LeftWheel_PWM,-858993460
LeftWheel_mVolts,1310 LeftWheel_Encoder,0 LeftWheel_PositionInMM,0 LeftWheel_RPM,-
13108 RightWheel_MaxPWM,65536 RightWheel_PWM,-858993460 RightWheel_mVolts,1310
RightWheel_Encoder,0 RightWheel_PositionInMM,0 RightWheel_RPM,-13108
Laser_MaxPWM,65536 Laser_PWM,-858993460 Laser_mVolts,1310 Laser_Encoder,0 Laser_RPM,
52428 Charger_MaxPWM,65536 Charger_PWM,-858993460 Charger_mAH, 52428
```

Command: GetSchedule

Description: Get the Cleaning Schedule. (24 hour clock format)
Usage: GetSchedule [Day <Day_value>]
Options:

Flag	Description

Day	Day of the week to get schedule for. Sun=0,Sat=6. If not specified, then all days are given.

Return Format:

```
Schedule is Enabled Sun 00:00 - None - Mon 00:00 - None - Tue 00:00 R Wed 00:00 R
Thu 00:00 R Fri 00:00 H Sat 00:00 H
```

Command: GetSysLog

Description: Get System Log data.
Usage: GetSysLog
Options: None
Return Format:

```
(Unimplemented) Sys Log Entries: Run, Stat, Min, Max, Sum, Count, Time(ms)
```

Command: GetTime

Description: Get Current Scheduler Time.
Usage: GetTime
Options: None
Return Format:

```
DayOfWeek HourOf24:Min:Sec Example: Sunday 13:57:09
```

Command: GetVersion

Description: Get the version information for the system software and hardware.
Usage: GetVersion
Options: None
Return Format:

```
Component,Major,Minor,Build ModelID,0,XV11, ConfigID,1,, Serial
Number,AAAnnnnnAA,0000000,D Software,2,1,15499 BatteryType,1,NIMH_12CELL,
BlowerType,1,BLOWER_ORIG, BrushSpeed,0,, BrushMotorType,1,BRUSH_MOTOR_ORIG,
SideBrushType,1,SIDE_BRUSH_NONE, WheelPodType,1,WHEEL_POD_ORIG,
DropSensorType,1,DROP_SENSOR_ORIG, MagSensorType,1,MAG_SENSOR_ORIG,
WallSensorType,1,WALL_SENSOR_ORIG, Locale,1,LOCALE_USA, LDS Software,V1.0.0,, LDS
Serial,XXX-YYY,, LDS CPU,F2802x/cd00,, MainBoard Vendor ID,1,, MainBoard Serial
Number,99,, MainBoard Version,15,0, ChassisRev,-1,, UIPanelRev,-1,,
```

Command: GetWarranty

Description: Get the warranty validation codes.
Usage: GetWarranty
Options: None
Return Format:

```
00000000 0000 962d3a58
```

Command: Help

Description: Without any argument, this prints a list of all possible cmds.
With a command name, it prints the help for that particular command
Usage: Help [Cmd <Cmd_value>]
Options:

Flag	Description
Cmd	(Optional) Next argument is command to show help for. If Cmd option not used, help gives list of all commands.

Return Format:

```
'Help' Generates: Help - Without any argument, this prints a list of all possible
cmds. With a command name, it prints the help for that particular command Clean -
Starts a cleaning by simulating press of start button. DiagTest - Executes different
test modes. Once set, press Start button to engage.
(Test modes are mutually exclusive.) GetAccel - Get the Accelerometer readings.
GetAnalogSensors - Get the A2D readings for the analog sensors. GetButtons - Get the
state of the UI Buttons. GetCalInfo - Prints out the cal info from the System
Control Block. GetCharger - Get the diagnostic data for the charging system.
GetDigitalSensors - Get the state of the digital sensors. GetErr - Get Error
Message. GetLDSScan - Get scan packet from LDS. GetLifeStatLog - Get All Life Stat
Logs. GetMotors - Get the diagnostic data for the motors. GetSchedule - Get the
Cleaning Schedule. (24 hour clock format) GetSysLog - Get System Log data. GetTime -
Get Current Scheduler Time. GetVersion - Get the version information for the system
software and hardware. GetWarranty - Get the warranty validation codes. PlaySound -
Play the specified sound in the robot. RestoreDefaults - Restore user settings to
default. SetDistanceCal - Set distance sensor calibration values for min and max
distances. SetFuelGauge - Set Fuel Gauge Level. SetMotor - Sets the specified motor
to run in a direction at a requested speed. (TestMode Only) SetTime - Sets the
current day, hour, and minute for the scheduler clock. SetLED - Sets the specified
LED to on,off,blink, or dim. (TestMode Only) SetLCD - Sets the LCD to the specified
display. (TestMode Only) SetLDSRotation - Sets LDS rotation on or off. Can only be
run in TestMode. SetSchedule - Modify Cleaning Schedule. SetSystemMode - Set the
operation mode of the robot. (TestMode Only) SetWallFollower - Enables/Disables wall
follower TestMode - Sets TestMode on or off. Some commands can only be run in
TestMode. Upload - Uploads new program to the robot.  'Help Clean' Generates: Clean
- Starts a cleaning by simulating press of start button.     House - (Optional)
Equivalent to pressing 'Start' button once. Starts a house cleaning. (House cleaning
mode is the default cleaning mode.)     Spot - (Optional) Starts a spot clean.
Stop - Stop Cleaning.
```

Command: PlaySound

Description: Play the specified sound in the robot.
Usage: PlaySound [SoundID <SoundID_value>] [Stop]
Options:

Flag	Description
SoundID	Play the sound library entry specified by the number in the next argument. Legal values are:

	0 – Waking Up 1 – Starting Cleaning 2 – Cleaning Completed 3 – Attention Needed 4 – Backing up into base station 5 – Base Station Docking Completed 6 – Test Sound 1 7 – Test Sound 2 8 – Test Sound 3 9 – Test Sound 4 10 – Test Sound 5 11 – Exploring 12 – ShutDown 13 – Picked Up 14 – Going to sleep 15 – Returning Home 16 – User Canceled Cleaning 17 – User Terminated Cleaning 18 – Slipped Off Base While Charging 19 – Alert 20 – Thank You
Stop	Stop playing sound.

Return Format: None

Command: RestoreDefaults

Description: Restore user settings to default.
Usage: RestoreDefaults
Options: None
Return Format: None

Command: SetDistanceCal

Description: Set distance sensor calibration values for min and max distances.
Usage: SetDistanceCal [DropMinimum] [DropMiddle] [DropMaximum] [WallMinimum] [WallMiddle] [WallMaximum]
Options:

Flag	Description
DropMinimum	Take minimum distance drop sensor readings. Mutually exclusive of DropMiddle and DropMax.
DropMiddle	Take middle distance drop sensor readings. Mutually exclusive of DropMinimim and DropMax.
DropMaximum	Take maximum distance drop sensor readings. Mutually exclusive of DropMinimum and DropMiddle.
WallMinimum	Take minimum distance wall sensor readings. Mutually exclusive of WallMiddle and WallMax.
WallMiddle	Take middle distance wall sensor readings. Mutually exclusive of

	WallMinimim and WallMax.
WallMaximum	Take maximum distance wall sensor readings. Mutually exclusive of WallMinimim and WallMiddle.

Return Format:

```
Label,Value RDropCalA2DMin,-1 RDropCalA2DMid,-1 RDropCalA2DMax,-1 LDropCalA2DMin,-1
LDropCalA2DMid,-1 LDropCalA2DMax,-1 WallCalA2DMin,-1 WallCalA2DMid,-1
WallCalA2DMax,-1
```

Command: SetFuelGauge

Description: Set Fuel Gauge Level.
Usage: SetFuelGauge [Percent <Percent_value>]
Options:

Flag	Description
Percent	Fuel Gauge percent from 0 to 100

Return Format: None

Command: SetLCD

Description: Sets the LCD to the specified display. (TestMode Only)
Usage: SetLCD [BGWhite] [BGBlack] [HLine <HLine_value>] [VLine <VLine_value>] [HBars] [VBars] [FGWhite] [FGBlack] [Contrast <Contrast_value>]
Options:

Flag	Description
BGWhite	Fill LCD background with White
BGBlack	Fill LCD background with Black
HLine	Draw a horizontal line (in foreground color) at the following row.
VLine	Draw a vertical line (in foreground color) at the following column.
HBars	Draw alternating horizontal lines (FG,BG,FG,BG,...), across the whole screen.
VBars	Draw alternating vertical lines (FG,BG,FG,BG,...), across the whole screen.
FGWhite	Use White as Foreground (line) color
FGBlack	Use Black as Foreground (line) color
Contrast	Set the following value as the LCD Contrast value into NAND. 0..63

Return Format: None

Command: SetLDSRotation

Description: Sets LDS rotation on or off. Can only be run in TestMode.
Usage: SetLDSRotation [On] [Off]
Options:

Flag	Description
On	Turns LDS rotation on. Mutually exclusive with Off.
Off	Turns LDS rotation off. Mutually exclusive with On.

Return Format: None

Command: SetLED

Description: Sets the specified LED to on,off,blink, or dim. (TestMode Only)
Usage: SetLED [BacklightOn] [BacklightOff] [ButtonAmber] [ButtonGreen] [LEDRed] [LEDGreen] [ButtonAmberDim] [ButtonGreenDim] [ButtonOff]
Options:

Flag	Description
BacklightOn	LCD Backlight On (mutually exclusive of BacklightOff)
BacklightOff	LCD Backlight Off (mutually exclusive of BacklightOn)
ButtonAmber	Start Button Amber (mutually exclusive of other Button options)
ButtonGreen	Start Button Green (mutually exclusive of other Button options)
LEDRed	Start Red LED (mutually exclusive of other Button options)
LEDGreen	Start Green LED (mutually exclusive of other Button options)
ButtonAmberDim	Start Button Amber Dim (mutually exclusive of other Button options)
ButtonGreenDim	Start Button Green Dim (mutually exclusive of other Button options)
ButtonOff	Start Button Off

Return Format: None

Command: SetMotor

Description: Sets the specified motor to run in a direction at a requested speed. (TestMode Only)
Usage: SetMotor [LWheelDist <LWheelDist_value>] [RWheelDist <RWheelDist_value>] [Speed <Speed_value>] [Accel <Accel_value>] [RPM <RPM_value>] [Brush] [VacuumOn] [VacuumOff] [VacuumSpeed <VacuumSpeed_value>] [RWheelDisable] [LWheelDisable] [BrushDisable] [RWheelEnable] [LWheelEnable] [BrushEnable]
Options:

Flag	Description
LWheelDist	Distance in millimeters to drive Left wheel. (Pos = forward, neg = backward)
RWheelDist	Distance in millimeters to drive Right wheel. (Pos = forward, neg = backward)
Speed	Speed in millimeters/second. (Required only for wheel movements)
Accel	Acceleration in millimeters/second. (Used only for wheel movements. Defaults to 'Speed'.)
RPM	Next argument is the RPM of the motor. Not used for wheels, but applied to all other motors specified in the command line.
Brush	Brush motor forward (Mutually exclusive with wheels and vacuum.)

VacuumOn	Vacuum motor on (Mutually exclusive with VacuumOff)
VacuumOff	Vacuum motor off (Mutually exclusive with VacuumOn)
VacuumSpeed	Vacuum speed in percent (1-100).
RWheelDisable	Disable Right Wheel motor
LWheelDisable	Disable Left Wheel motor
BrushDisable	Disable Brush motor
RWheelEnable	Enable Right Wheel motor
LWheelEnable	Enable Left Wheel motor
BrushEnable	Enable Brush motor

Return Format: None

Command: SetSchedule

Description: Modify Cleaning Schedule.
Usage: SetSchedule [Day <Day_value>] [Hour <Hour_value>] [Min <Min_value>] [House] [None] [ON] [OFF]
Options:

Flag	Description
Day	Day of the week to schedule cleaning for. Sun=0,Sat=6. (required)
Hour	Hour value 0..23 (required)
Min	Minutes value 0..59 (required)
House	Schedule to Clean whole house (default) (Mutually exclusive with None)
None	Remove Scheduled Cleaning for specified day. Time is ignored. (Mutually exclusive with House)
ON	Enable Scheduled cleanings (Mutually exclusive with OFF)
OFF	Disable Scheduled cleanings (Mutually exclusive with ON)

Return Format:

```
Schedule is Enabled Sun 00:00 - None - Mon 00:00 - None - Tue 00:00 R Wed 00:00 R
Thu 00:00 R Fri 00:00 H Sat 00:00 H
```

Command: SetSystemMode

Description: Set the operation mode of the robot. (TestMode Only)
Usage: SetSystemMode [Shutdown] [Hibernate] [Standby]
Options:

Flag	Description
Shutdown	Shut down the robot. (mutually exclusive of other options)
Hibernate	Start hibernate operation.(mutually exclusive of other options)

Standby	Start standby operation. (mutually exclusive of other options)

Return Format: None

Command: SetTime

Description: Sets the current day, hour, and minute for the scheduler clock.
Usage: SetTime [Day <Day_value>] [Hour <Hour_value>] [Min <Min_value>] [Sec <Sec_value>]
Options:

Flag	Description
Day	Day of week value Sunday=0,Monday=1,... (required)
Hour	Hour value 0..23 (required)
Min	Minutes value 0..59 (required)
Sec	Seconds value 0..59 (Optional, defaults to 0)

Return Format: None

Command: SetWallFollower

Description: Enables/Disables wall follower
Usage: SetWallFollower [Enable] [Disable]
Options:

Flag	Description
Enable	Enable wall follower
Disable	Disable wall follower

Return Format: None

Command: TestMode

Description: Sets TestMode on or off. Some commands can only be run in TestMode.
Usage: TestMode [On] [Off]
Options:

Flag	Description
On	Turns Testmode on. Mutually exclusive with Off.
Off	Turns Testmode off. Mutually exclusive with On.

Return Format: None

Command: Upload

Description: Uploads new program to the robot.
Usage: Upload [dump] [code] [sound] [LDS] [xmodem] [size <size_value>] [noburn] [readflash] [reboot]
Options:

Flag	Description
dump	Dump the contents of the upload save area.

code	Upload file is the main application. (Mutually exclusive with sound, LDS)
sound	Upload file is a sound module. (Mutually exclusive with code, LDS)
LDS	Upload file is an LDS module. (Mutually exclusive with sound, code)
xmodem	Use xmodem protocal when transmitting upload module
size	data size to upload to device.
noburn	test option -- do NOT burn the flash after the upload.
readflash	test option -- read the flash at the current region.
reboot	Reset the robot after performing the upload.

Return Format: None

Anexo B

Código Python (versión Socket)

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\neatoSRV\neatoSRV.py miércoles, 21 de octubre de 2015 21:10

```
'''
    Python code for serial and socket communication between a neato device and a socket client
'''
import socket
import sys
import time
import timeit
from NeatoSerialInterface import NeatoSerialInterface
#-----------------------
if sys.platform == 'win32':
    import serial.tools.list_ports # Getting serial port number in windows
    from thread import *
    from Queue import Queue
else:
    from multiprocessing import Process, Queue

# Incicializacion de variables
HOST = ''    # Vacio -> Se publica en todas las interficies
PORT = 20000 # Arbitrary non-privileged port
q_cli2neato = Queue() # put(), empty(), get()
q_neato2cli = Queue()
q_stop = Queue()
start = time.time() # Guardamos tiempo inicial para timestamp (no se modifica)
session = False # Indica si hay una session activa
data = '' # Necesario para saber si se para el bucle principal o no
mode = '' # Indica el modo de funcionamiento
#=============================================================
#device = '/dev/ttyACM0'              # Raspbian
#device = '/dev/tty.usbmodem1d131'    # Mac OS X
#device = 'DEBUG'                     # Debugging
#device = 'COM4'                      # Windows
#=============================================================
device = 'null'
timeout_sec = 0.13
lidar_freq = 0.0

#Recibe los comandos de la aplicacion por el socket
def sckt_lstn_thread(conn,q_cli2neato):
    # Iniciando variables locales
    global start

    recv = ""
    cmm = ""
    #Bucle que trata la conexion con el cliente ya conectado
    while True:
        #Receiving from client
        recv += conn.recv(1024) # dat incorpora salto de linea al final #Es bloqueante
        (2048) | Acaba en 13 10

        if chr(10) in recv:
            cmm,recv = recv.split(chr(10),1) #Se separa el primer comando antes del salto

        cmm = cmm.replace(chr(10),"") # Se elimina el salto de linea - NL
        cmm = cmm.replace(chr(13),"") # Se elimina el salto de linea - CR
        timestamp = time.time() - start # Obtenemos timestamp
        ts = str(timestamp)
        print 'SktLst - ' + ts + ' | ' + cmm # Imprime el mensaje recibido con el timestamp
```

```
        # Colocamos la info en la queue
        q_cli2neato.put(cmm)

        # Finalizacion del bucle
        if cmm == 'Stop' or cmm == 'Close':
            break

#Envia las respuestas a la aplicacion por el socket
def sckt_wrt_thread(conn,q_neato2cli):
    global session
    global start
    lopp = True

    while lopp:
        while q_neato2cli.empty():
            pass

        while not q_neato2cli.empty():
            resp = q_neato2cli.get()
            if resp == 'Stop' or resp == 'Close':
                session = False
                lopp = False
                conn.send('Sistema: ' + resp)
                conn.close()
                break
            else:
                conn.send(resp + chr(10))
    sys.exit()

################################################################################
def check_command(q_cli2neato,q_neato2cli,q_stop,neato):
    global start
    loop = True

    if not q_cli2neato.empty():
        while not q_cli2neato.empty(): #Se vacian todos los comandos pendientes
            dat = q_cli2neato.get()

            if dat == 'Stop' or dat == 'Close':
                timestamp = time.time() - start # Obtenemos timestamp
                print 'Serial - ' + str(timestamp) + ' | ' + dat
                q_neato2cli.put(dat);
                neato.Stop()
                loop = False
                q_stop.put(dat); # Indica el Stop o el Close al main
                break
            resp = neato.getCmmd(dat)
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' | ' + resp
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#")
            q_neato2cli.put(send) # pasamos info al cliente
    return loop
```

```
################################################################################
def serial_thread_listener(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global lidar_freq
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        while loop:
            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

################################################################################
def serial_thread_constant(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global lidar_freq
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        while loop:
            # Run scan command
            # Enviar todo
            info = neato.getAll()
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' # ' + info
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#")
            q_neato2cli.put(send) # pasamos info al cliente
```

```
            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

##############################################################################################
def serial_thread_moody(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global lidar_freq
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        it=0; # Indica el numero de iteracion para saber que dato se debe pedir, (enc o todo)
        while loop:
            # Run scan command
            if(it < 2) : # Enviar motores
                resp = neato.getCmmd('GetMotors LeftWheel RightWheel')
                it += 1;
            else: # Enviar todo
                resp = neato.getAll()
                it = 0;
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' # ' + resp
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#")
            q_neato2cli.put(send) # pasamos info al cliente

            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

##############################################################################################
def get_serial_port():
    puerto = 'null'
    if sys.platform == 'win32': # Si plataforma windows
        ports = list(serial.tools.list_ports.comports())
        for p in ports:
            #print p
            if 'Neato' in p[1]:
                puerto = p[0]
    else: # Si plataforma linux (Raspbian)
        puerto = '/dev/ttyACM0'
    return puerto
```

```
#================================== MAIN ================================================
if __name__ == '__main__':
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print 'Socket created'

    #Bind socket to local host and port
    try:
        s.bind((HOST, PORT))
    except socket.error as msg:
        print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + str(msg[1])
        sys.exit()

    print 'Socket bind complete'
    loop = True
    q_stop.put("first") # Se pone dato para saltar el bucle de espera inicial

    #Start listening on socket
    s.listen(1)
    print 'Socket now listening'

    # Bucle principal, escucha conexiones entrantes y genera threads cuando hay conexion
    while loop:

        # Espera a que se elimine la conexion existente
        while q_stop.empty():
            pass

        # Ha llegado instruccion a "q_stop"
        dat = q_stop.get()
        # Revisa si hay que seguir aceptando conexiones o se sale
        if dat == 'Stop': # Parar el programa
            print 'Main Stop'
            loop = False
            s.close()
            time.sleep(2)
            sys.stdout.flush()
            sys.exit()
            break
        if data == 'Close':
            data = ''
            break

        print '\nWaiting for a connection...\n'

        # Espera a establecer conexion - llamada bloqueante
        conn, addr = s.accept()
        #start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp
        print 'Connected with ' + addr[0] + ':' + str(addr[1])

        #Enviar mensaje sobre la deteccion del puerto serie (needed, handshaking)
        device = get_serial_port()

        run = False
        if device != 'null':
```

```
            run = True
            conn.send('Serial OK' + chr(10))
        else:
            conn.send('Serial KO' + chr(10))
            loop = False
            s.close()
            time.sleep(2)
            sys.stdout.flush()
            sys.exit()

        if run:
            # Esperar mensaje de modo de funcionamiento (DataMode [listener,constant,moody])
            (needed, handshaking)
            repeat = True
            while repeat:
                dat = conn.recv(1024) # dat incorpora salto de linea al final #Es bloqueante
                (2048)
                print dat
                dat = dat.replace(chr(26),"")
                dat = dat.replace(chr(10),"")
                mod = dat.split()
                if (mod[0] == 'DataMode') and ( mod[1] == 'listener' or mod[1] ==
                'constant'or mod[1] == 'moody'):
                    repeat = False
                    conn.send(mod[0] + ' '+ mod[1]+ ' OK' + chr(10))
                    mode = mod[1]

            # Se generan nuevos threads, primer argumento, nombre de la func a ejecutar
            # segundo, tupla con los argumentos de la funcion
            # Se diferencia entre windows y linux, en windows threads virtuales, en windows
            multihreading real
            if sys.platform == 'win32':
            #### Threads virtuales #####
                # Thread que escucha el socket y encola los mensajes entrantes desde la
                aplicacion
                if mode == 'listener':
                    start_new_thread(serial_thread_listener ,(q_cli2neato,q_neato2cli,q_stop))
                elif mode == 'constant':
                    start_new_thread(serial_thread_constant ,(q_cli2neato,q_neato2cli,q_stop))
                else: #mode == moody
                    start_new_thread(serial_thread_moody ,(q_cli2neato,q_neato2cli,q_stop))
                time.sleep(3.5) # Esperar a que el thread serie se estabilice, usar queue
                para sincronizar???
                start_new_thread(sckt_lstn_thread ,(conn,q_cli2neato))
                start_new_thread(sckt_wrt_thread ,(conn,q_neato2cli))
            else:
            #### Multithreading #####
                # Thread que escucha el socket y encola los mensajes entrantes desde la
                aplicacion
                if mode == 'listener':
                    ser = Process(target=serial_thread_listener,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
                elif mode == 'constant':
                    ser = Process(target=serial_thread_constant,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
```

```
                else: #mode == moody
                    ser = Process(target=serial_thread_moody,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
                time.sleep(3.5)
                sckt_lstn = Process(target=sckt_lstn_thread, args=(conn,q_cli2neato))
                sckt_wrt = Process(target=sckt_wrt_thread, args=(conn,q_neato2cli))
                sckt_lstn.start()
                sckt_wrt.start()

                ser.join()
                sckt_lstn.join()
                sckt_wrt.join()

    s.close()
    sys.exit()
```

```
#!/usr/bin/env python
'''
Based on:
  neatopylot_server.py - server code for Neato XV-11 Autopylot
  Copyright (C) 2013 Suraj Bajracharya and Simon D. Levy
-----------------------------------------------------------------
Esta clase realiza la conexion serie con el robot Neato y sus funciones son:
 - Enviar comandos al robot
 - Recibir datos del robot
'''
import serial
import time
import timeit
import sys

import io

class NeatoSerialInterface:
    def __init__(self, device, timeout): # mode -> indica si se trabaja con timeout o no
#el modo timeout envia todos los datos en el buffer una vez pasado el timeout
        try:
            #timeout = 0 -> no se espera el timeout
            self.robot = serial.Serial(device, 115200, serial.EIGHTBITS, serial.PARITY_NONE,
            serial.STOPBITS_ONE, timeout)
        except serial.SerialException as e:
            #Lanzar excepcion
            print('Unable to connect to XV-11 as device ' + device)
            print('Make sure XV-11 is powered on and USB cable is plugged in!')
            print("Exception({0}): {1}".format(e.errno, e.strerror))
            self.robot = False

    # Inicia el modo Test y enciende e Lidar
    def Start(self):
        print self.getbCmmd('TestMode on')
        print self.getbCmmd('SetLDSRotation on')
        print('Waiting scanner to be ready...')
        time.sleep(3) #Wait n seconds to the laser engine
        print('Scanner ready')

    # Envia comando finalizado en salto de linea y lee el resultado
    def getCmmd(self,command):
        #Envia comando al Neato
        self.robot.write(command + chr(10))
        self.robot.flush()
        #Devuelve respuesta
        return self.step_read()

    def getbCmmd(self,command):
        #Envia comando al Neato
        self.robot.write(command + chr(10))
        self.robot.flush()
        #Devuelve respuesta
        return self.blind_read()

        # Envia un comando solicitando todos los parametros disponibles (lectura por tam de
        buffer y timeout)
    def getAll(self):
```

```
        self.robot.write('GetMotors LeftWheel RightWheel' + chr(10) + 'GetAccel' + chr(10) +
        'GetAnalogSensors' + chr(10) + 'GetDigitalSensors' + chr(10) +'GetLDSScan' + chr(10))
        self.robot.flush()
        return self.step_readn(5)

    def step_read(self):
        end1 = chr(13) # CR - Carriage Return
        end2 = chr(10) # LF - New line
        end3 = chr(26) # SUB - Substitute
        line = bytearray()
        p1 = self.robot.read(1)
        p2 = self.robot.read(1)
        p3 = self.robot.read(1)
        if p1 and p2 and p3 :
            while True:
                if p1 == end1 and p2 == end2 and p3 == end3:
                    break
                line += p1
                p1 = p2
                p2 = p3
                p3 = self.robot.read(1)
        return bytes(line)


    def blind_read(self):
        line = bytearray()
        p = self.robot.read(1)
        if p:
            while True:
                line += p
                p = self.robot.read(1)
                if not p:
                    break
        return bytes(line)


    def step_readn(self,n):
        end1 = chr(13) # CR - Carriage Return
        end2 = chr(10) # LF - New line
        end3 = chr(26) # SUB - Substitute
        line = bytearray()
        p1 = self.robot.read(1)
        p2 = self.robot.read(1)
        p3 = self.robot.read(1)
        if p1 and p2 and p3 :
            while True:
                if p1 == end1 and p2 == end2 and p3 == end3:
                    n = n-1
                    if n == 0:
                        break
                line += p1
                p1 = p2
                p2 = p3
                p3 = self.robot.read(1)
                if not p3:
                    break
        return bytes(line)
```

```
    # Detiene modo test y Lidar, y finaliza la conexion serie
    def Stop(self):
        # Shut down the XV-11
        print('Shutting down ...')
        # Spin down the LIDAR
        print self.getbCmmd('SetLDSRotation off')
        # Take the XV-11 out of test mode
        print self.getbCmmd('TestMode off')
        # Close the port
        self.robot.close()
```

Anexo C

Código Python (versión WebSocket)

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\neatoWSRV\NeatoWebSRV.py miércoles, 21 de octubre de 2015 22:11

```
'''
    Python code for serial and socket communication between a neato device and a socket client
'''
import socket
import sys
import time
import timeit
from NeatoSerialInterface import NeatoSerialInterface
#-----------------------
from BaseHTTPServer import BaseHTTPRequestHandler
import struct
import hashlib
import base64
import SocketServer
from StringIO import StringIO
import codecs
#-----------------------
if sys.platform == 'win32':
    import serial.tools.list_ports # Getting serial port number in windows
    from thread import * # Test threading desde consola
    from Queue import Queue # Test threading desde consola
else:
    from multiprocessing import Process, Queue

# Incicializacion de variables
HOST = ''    # Vacio -> Se publica en todas las interficies
PORT = 20000 # Arbitrary non-privileged port
q_cli2neato = Queue() # put(), empty(), get()
q_neato2cli = Queue()
q_stop = Queue()
start = time.time() # Guardamos tiempo inicial para timestamp (no se modifica)
session = False # Indica si hay una session activa
data = '' # Necesario para saber si se para el bucle principal o no
mode = '' # Indica el modo de funcionamiento
#=============================================================
STREAM = 0x0
TEXT = 0x1
BINARY = 0x2
CLOSE = 0x8
PING = 0x9
PONG = 0xA

HEADERB1 = 1
HEADERB2 = 3
LENGTHSHORT = 4
LENGTHLONG = 5
MASK = 6
PAYLOAD = 7
#=============================================================
#device = '/dev/ttyACM0'              # Raspbian
#device = '/dev/tty.usbmodem1d131'    # Mac OS X
#device = 'DEBUG'                     # Debugging
#device = 'COM4'                      # Windows
#=============================================================
device = 'null'
# Serial-port timeout
timeout_sec = 0.13 # Para puerto serie
```

```
#Recibe los comandos de la aplicacion por el socket
def sckt_lstn_thread(conn,q_cli2neato):
    # Iniciando variables locales
    global start

    recv = ""
    cmm = ""
    #Bucle que trata la conexion con el cliente ya conectado
    while True:
        cmm = get_Data(conn)

        cmm = cmm.replace(chr(10),"") # Se elimina el salto de linea - NL
        cmm = cmm.replace(chr(13),"") # Se elimina el salto de linea - CR
        timestamp = time.time() - start # Obtenemos timestamp
        ts = str(timestamp)
        print 'SktLst - ' + ts + ' | ' + cmm # Imprime el mensaje recibido con el timestamp

        # Colocamos la info en la queue
        q_cli2neato.put(cmm)

        # Finalizacion del bucle
        if cmm == 'Stop' or cmm == 'Close':
            break

#Envia las respuestas a la aplicacion por el socket
def sckt_wrt_thread(conn,q_neato2cli):
    global session
    global start
    lopp = True

    while lopp:
        #start_time = timeit.default_timer() #tiempo de ejecucion

        while q_neato2cli.empty():
            pass

        while not q_neato2cli.empty():
            resp = q_neato2cli.get()
            if resp == 'Stop' or resp == 'Close':
                session = False
                lopp = False
                conn.send('Sistema: ' + resp)
                conn.close()
                break
            else:
                sendMessage(conn, resp) # +chr(10)

    sys.exit()
################################################################################
def check_command(q_cli2neato,q_neato2cli,q_stop,neato):
    global start
    loop = True

    if not q_cli2neato.empty():
        while not q_cli2neato.empty(): #Se vacian todos los comandos pendientes
            dat = q_cli2neato.get()
```

```
            if dat == 'Stop' or dat == 'Close':
                timestamp = time.time() - start # Obtenemos timestamp
                print 'Serial - ' + str(timestamp) + ' | ' + dat
                q_neato2cli.put(dat);
                neato.Stop()
                loop = False
                q_stop.put(dat); # Indica el Stop o el Close al main
                break
            print 'Serial getCmmd: ' + dat
            resp = neato.getCmmd(dat)
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' | ' + resp
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#")
            q_neato2cli.put(send) # pasamos info al cliente
    return loop

################################################################################
def serial_thread_listener(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        while loop:
            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

################################################################################
def serial_thread_constant(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
```

```
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        while loop:
            # Run scan command
            # Enviar todo
            info = neato.getAll()
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' # ' + info
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#") #(Revisar chr(26)-> SUB, reemplazar por #?)
            q_neato2cli.put(send) # pasamos info al cliente

            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

##############################################################################
def serial_thread_moody(q_cli2neato,q_neato2cli,q_stop):
    global start
    global session
    global device
    global timeout_sec

    loop = True

    neato = NeatoSerialInterface(device, timeout_sec)
    if type(neato.robot) is bool:
        dat = ''
        session = False
    else:
        neato.Start()
        start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp

        it=0; # Indica el numero de iteracion para saber que dato se debe pedir, (enc o todo)
        while loop:
            # Run scan command
            if(it < 2) : # Enviar motores
                resp = neato.getCmmd('GetMotors LeftWheel RightWheel')
                it += 1;
            else: # Enviar todo
                resp = neato.getAll()
                it = 0;
            timestamp = time.time() - start # Obtenemos timestamp
            send = str(timestamp) + ' # ' + resp
            print 'Serial - ' + send
            #Se eliminan los saltos de linea para poder enviar por socket
            send = send.replace(chr(13),";")
            send = send.replace(chr(10),"")
            send = send.replace(chr(26),"#")
            q_neato2cli.put(send) # pasamos info al cliente
```

```
            #revisamos si hay algun comando desde el cliente
            loop = check_command(q_cli2neato,q_neato2cli,q_stop,neato)
    sys.exit()

##############################################################################
def get_serial_port():
    puerto = 'null'
    if sys.platform == 'win32': # Windows platform
        ports = list(serial.tools.list_ports.comports())
        for p in ports:
            #print p
            if 'Neato' in p[1]:
                puerto = p[0]
    else: # Linux platform (Raspbian)
        puerto = '/dev/ttyACM0'
    return puerto


##############################################################################
#################### Websockket Specific Functions ###########################
##############################################################################
class HTTPRequest(BaseHTTPRequestHandler):
    def __init__(self, request_text):
        self.rfile = StringIO(request_text)
        self.raw_requestline = self.rfile.readline()
        self.error_code = self.error_message = None
        self.parse_request()


def do_handshaking(socket):
    headerbuffer = ''
    handshaked = False
    handshake_str = (
        "HTTP/1.1 101 Switching Protocols\r\n"
        "Upgrade: WebSocket\r\n"
        "Connection: Upgrade\r\n"
        "Sec-WebSocket-Accept: %(acceptstr)s\r\n\r\n"
    )
    guid_str = '258EAFA5-E914-47DA-95CA-C5AB0DC85B11'

    while not handshaked:
        # accumulate
        headerbuffer += socket.recv(2048) #2048 header to read

        if len(headerbuffer) >= 65536: # 65536 ->  max header
            raise Exception('header exceeded allowable size')

        # indicates end of HTTP header
        if '\r\n\r\n' in headerbuffer:
            request = HTTPRequest(headerbuffer)

            # handshake rfc 6455
            if request.headers.has_key('Sec-WebSocket-Key'.lower()):
                key = request.headers['Sec-WebSocket-Key'.lower()]
                hStr = handshake_str % { 'acceptstr' :  base64.b64encode(hashlib.sha1(key +
                guid_str).digest()) }
```

```
                ws_sendMessage(False, BINARY, hStr, socket)
                handshaked = True
            else:
                raise Exception('Sec-WebSocket-Key does not exist')
    return handshaked



def get_Data(socket):
    dat = socket.recv(8192) #8192,3072
    if not dat:
        raise Exception("remote socket closed")
    #for d in dat:
    return ws_parseMessage(dat, socket) # retorna solo el mensaje o None


def ws_parseMessage(dat, socket):
    state = HEADERB1
    fin = 0
    opcode = 0
    index = 0
    mlength = None
    lengtharray = None
    data = bytearray()
    ret = None
    hasmask = 0
    maskarray = None

    frag_type = BINARY
    frag_start = False
    frag_buffer = None

    for d in dat:
        byte = ord(d)

        # read in the header
        if state == HEADERB1:
            fin = byte & 0x80
            opcode = byte & 0x0F
            state = HEADERB2

            index = 0
            mlength = 0
            lengtharray = bytearray()
            data = bytearray()

            rsv = byte & 0x70
            if rsv != 0:
                raise Exception('RSV bit must be 0')

        elif state == HEADERB2:
            mask = byte & 0x80
            length = byte & 0x7F

            if opcode == PING and length > 125:
                raise Exception('ping packet is too large')
```

```
            if mask == 128:
                hasmask = True
            else:
                hasmask = False

            if length <= 125:
                mlength= length

                # if we have a mask we must read it
                if hasmask is True:
                    maskarray = bytearray()
                    state = MASK
                else:
                    # if there is no mask and no payload we are done
                    if mlength<= 0:
                        try:
                            opcode, data, fin, frag_type, frag_start, frag_buffer, dataok =
                            ws_handlePacket(socket, opcode, data, fin, frag_type,
                            frag_start, frag_buffer)
                            if dataok:
                                ret = data
                        finally:
                            state = HEADERB1
                            data = bytearray()
                    # we have no mask and some payload
                    else:
                        #self.index = 0
                        data = bytearray()
                        state = PAYLOAD
            elif length == 126:
                lengtharray = bytearray()
                state = LENGTHSHORT
            elif length == 127:
                lengtharray = bytearray()
                state = LENGTHLONG

        elif state == LENGTHSHORT:
            lengtharray.append(byte)

            if len(lengtharray) > 2:
                raise Exception('short length exceeded allowable size')

            if len(lengtharray) == 2:
                mlength= struct.unpack_from('!H', str(lengtharray))[0]

                if hasmask is True:
                    maskarray = bytearray()
                    state = MASK
                else:
                    # if there is no mask and no payload we are done
                    if mlength <= 0:
                        try:
                            opcode, data, fin, frag_type, frag_start, frag_buffer, dataok =
                            ws_handlePacket(socket, opcode, data, fin, frag_type,
                            frag_start, frag_buffer)
                            if dataok:
                                ret = data
```

```
                    finally:
                        state = HEADERB1
                        data = bytearray()
                # we have no mask and some payload
                else:
                    data = bytearray()
                    state = PAYLOAD

        elif state == LENGTHLONG:
            lengtharray.append(byte)
            if len(lengtharray) > 8:
                raise Exception('long length exceeded allowable size')

            if len(lengtharray) == 8:
                mlength= struct.unpack_from('!Q', str(lengtharray))[0]

                if hasmask is True:
                    maskarray = bytearray()
                    state = MASK
                else:
                    # if there is no mask and no payload we are done
                    if mlength <= 0:
                        try:
                            opcode, data, fin, frag_type, frag_start, frag_buffer, dataok =
                            ws_handlePacket(socket, opcode, data, fin, frag_type,
                            frag_start, frag_buffer)
                            if dataok:
                                ret = data
                        finally:
                            state = HEADERB1
                            data = bytearray()
                    # we have no mask and some payload
                    else:
                        data = bytearray()
                        state = PAYLOAD

        # MASK STATE
        elif state == MASK:
            maskarray.append(byte)
            if len(maskarray) > 4:
                raise Exception('mask exceeded allowable size')

            if len(maskarray) == 4:
                # if there is no mask and no payload we are done
                if mlength<= 0:
                    try:
                        opcode, data, fin, frag_type, frag_start, frag_buffer, dataok =
                        ws_handlePacket(socket, opcode, data, fin, frag_type, frag_start,
                        frag_buffer)
                        if dataok:
                            ret = data
                    finally:
                        state = HEADERB1
                        data = bytearray()
                    # we have no mask and some payload
                else:
                    data = bytearray()
```

```
                    state = PAYLOAD

        # PAYLOAD STATE
        elif state == PAYLOAD:
            if hasmask is True:
                data.append( byte ^ maskarray[index % 4] )
            else:
                data.append( byte )

            # if length exceeds allowable size then we except and remove the connection
            if len(data) >= 33554432: # Max payload
                raise Exception('payload exceeded allowable size')

            # check if we have processed length bytes; if so we are done
            if (index+1) == mlength:
                try:
                    opcode, data, fin, frag_type, frag_start, frag_buffer, dataok =
                    ws_handlePacket(socket, opcode, data, fin, frag_type, frag_start,
                    frag_buffer)
                    if dataok:
                        ret = data
                finally:
                    state = HEADERB1
                    data = bytearray()
            else:
                index += 1
    return ret


def ws_handlePacket(socket, opcode, data, fin, frag_type, frag_start, frag_buffer):
    dataok = False
    _VALID_STATUS_CODES = [1000, 1001, 1002, 1003, 1007, 1008, 1009, 1010, 1011, 3000, 3999,
    4000, 4999]
    frag_decoder = codecs.getincrementaldecoder('utf-8')(errors='strict')

    if opcode == CLOSE or opcode == STREAM or opcode == TEXT or opcode == BINARY:
        pass
    elif opcode == PONG or opcode == PING:
        if len(data) > 125:
            raise Exception('control frame length can not be > 125')
    else:
        # unknown or reserved opcode so just close
        raise Exception('unknown opcode')

    if opcode == CLOSE:
        status = 1000
        reason = u''
        length = len(data)

        if length == 0:
            pass
        elif length >= 2:
            status = struct.unpack_from('!H', data[:2])[0]
            reason = data[2:]

            if status not in _VALID_STATUS_CODES:
                status = 1002
```

```
            if len(reason) > 0:
                try:
                    reason = reason.decode('utf-8', errors='strict')
                except:
                    status = 1002
        else:
            status = 1002
        ws_close(socket, status, reason)
        return

    elif fin == 0:
        if opcode != STREAM:
            if opcode == PING or opcode == PONG:
                raise Exception('control messages can not be fragmented')

            frag_type = opcode
            frag_start = True
            frag_decoder.reset()

            if frag_type == TEXT:
                frag_buffer = []
                utf_str = frag_decoder.decode(data, final = False)
                if utf_str:
                    frag_buffer.append(utf_str)
            else:
                frag_buffer = bytearray()
                frag_buffer.extend(data)

        else:
            if frag_start is False:
                raise Exception('fragmentation protocol error')

            if frag_type == TEXT:
                utf_str = frag_decoder.decode(data, final = False)
                if utf_str:
                    frag_buffer.append(utf_str)
            else:
                frag_buffer.extend(data)

    else:
        if opcode == STREAM:
            if frag_start is False:
                raise Exception('fragmentation protocol error')

            if frag_type == TEXT:
                utf_str = frag_decoder.decode(data, final = True)
                frag_buffer.append(utf_str)
                data = u''.join(frag_buffer)
            else:
                frag_buffer.extend(data)
                data = frag_buffer


            frag_decoder.reset()
            frag_type = BINARY
            frag_start = False
```

```
            frag_buffer = None
            dataok = True

        elif opcode == PING:
            ws_sendMessage(False, PONG, data, socket)

        elif opcode == PONG:
            pass

        else:
            if frag_start is True:
                raise Exception('fragmentation protocol error')

            if opcode == TEXT:
                try:
                    data = data.decode('utf-8', errors='strict')
                except Exception as exp:
                    raise Exception('invalid utf-8 payload')

            dataok = True
    return opcode, data, fin, frag_type, frag_start, frag_buffer, dataok



def ws_close(socket, status = 1000, reason = u''):
    """
        Send Close frame to the client. The underlying socket is only closed
        when the client acknowledges the Close frame.

        status is the closing identifier.
        reason is the reason for the close.
    """

    close_msg = bytearray()
    close_msg.extend(struct.pack("!H", status))
    if isinstance(reason, unicode):
        close_msg.extend(reason.encode('utf-8'))
    else:
        close_msg.extend(reason)

    ws_sendMessage(False, CLOSE, str(close_msg),socket)



def sendMessage(socket, data):
    """
        Send websocket data frame to the client.
        If data is a unicode object then the frame is sent as Text.
        If the data is a bytearray object then the frame is sent as Binary.
    """
    opcode = BINARY
    if isinstance(data, unicode):
        opcode = TEXT
    ws_sendMessage(False, TEXT, data, socket)
```

```
def ws_sendMessage(fin, opcode, data, socket):
    header = bytearray()
    b1 = 0
    b2 = 0
    if fin is False:
        b1 |= 0x80
    b1 |= opcode

    if isinstance(data, unicode):
        data = data.encode('utf-8')

    length = len(data)
    header.append(b1)

    if length <= 125:
        b2 |= length
        header.append(b2)
    elif length >= 126 and length <= 65535:
        b2 |= 126
        header.append(b2)
        header.extend(struct.pack("!H", length))

    else:
        b2 |= 127
        header.append(b2)
        header.extend(struct.pack("!Q", length))

    payload = None
    if length > 0:
        payload = str(header) + str(data)
    else:
        payload = str(header)
    size = len(payload)
    tosend = size
    already_sent = 0
    while tosend > 0:
        try:
            # i should be able to send a bytearray
            sent = socket.send(payload[already_sent:])
            if sent == 0:
                raise RuntimeError("socket connection broken")
            already_sent += sent
            tosend -= sent
        except socket.error as e:
            print 'Exception i ws_sendMessage: '+ str(e.errno)


def testing_conn(socket):
    rec = None
    while rec != 'Stop':
        msg = '8.94099998474;^GetMotors LeftWheel
        RightWheel;Parameter,Value;LeftWheel_RPM,0;LeftWheel_Load%,0;LeftWheel_PositionInMM,0;
        LeftWheel_Speed,0;RightWheel_RPM,0;RightWheel_Load%,0;RightWheel_PositionInMM,0;RightW
        heel_Speed,0;^GetAccel;Label,Value;PitchInDegrees,  0.50;RollInDegrees, -0.95;XInG,
        0.009;YInG,-0.017;ZInG, 1.020;SumInG,
```

1.020;^GetAnalogSensors;SensorName,Value;WallSensorInMM,68,;BatteryVoltageInmV,15584,;
LeftDropInMM,60,;RightDropInMM,60,;LeftMagSensor,0,;RightMagSensor,-1,;UIButtonInmV,33
08,;VacuumCurrentInmA,0,;ChargeVoltInmV,421,;BatteryTemp0InC,28,;BatteryTemp1InC,26,;C
urrentInmA,537,;SideBrushCurrentInmA,0,;VoltageReferenceInmV,1224,;AccelXInmG,12,;Acce
lYInmG,-20,;AccelZInmG,1020,;^GetDigitalSensors;Digital Sensor Name,
Value;SNSR_DC_JACK_CONNECT,0;SNSR_DUSTBIN_IS_IN,1;SNSR_LEFT_WHEEL_EXTENDED,0;SNSR_RIGH
T_WHEEL_EXTENDED,0;LSIDEBIT,0;LFRONTBIT,0;RSIDEBIT,0;RFRONTBIT,0;^GetLDSScan;AngleInDe
grees,DistInMM,Intensity,ErrorCodeHEX;0,2785,81,0;1,2786,76,0;2,0,76,8021;3,2798,76,0;
4,2781,24,0;5,2153,27,0;6,2163,34,0;7,2169,29,0;8,2172,33,0;9,2222,31,0;10,2224,34,0;1
1,1811,6,0;12,0,0,8035;13,0,0,8035;14,0,0,8035;15,2931,10,0;16,0,0,8035;17,2935,41,0;1
8,2956,47,0;19,2969,45,0;20,2995,46,0;21,3011,45,0;22,3041,45,0;23,3058,44,0;24,3089,3
9,0;25,3112,40,0;26,3131,31,0;27,3158,34,0;28,3133,16,0;29,0,0,8035;30,0,0,8035;31,0,0
,8035;32,0,0,8035;33,0,0,8035;34,0,0,8035;35,3469,7,0;36,0,0,8035;37,0,0,8035;38,0,0,8
035;39,0,0,8035;40,0,0,8035;41,0,0,8035;42,0,0,8035;43,847,18,0;44,857,24,0;45,871,33,
0;46,886,70,0;47,903,17,0;48,919,87,0;49,0,0,8035;50,0,0,8021;51,992,16,0;52,3439,10,0
;53,0,0,8035;54,0,0,8035;55,0,0,8035;56,2058,16,0;57,0,0,8035;58,0,0,8035;59,0,0,8035;
60,0,0,8035;61,0,0,8035;62,0,0,8035;63,386,100,0;64,0,0,8002;65,360,130,0;66,349,217,0
;67,343,237,0;68,339,245,0;69,337,204,0;70,332,231,0;71,335,204,0;72,344,214,0;73,350,
318,0;74,352,277,0;75,353,332,0;76,353,334,0;77,354,464,0;78,354,297,0;79,353,194,0;80
,0,0,8035;81,0,0,8035;82,0,0,8035;83,0,0,8035;84,0,0,8035;85,0,0,8035;86,0,0,8035;87,0
,0,8002;88,0,0,8035;89,0,0,8035;90,0,0,8035;91,0,0,8035;92,0,0,8035;93,0,0,8035;94,143
0,9,0;95,0,0,8002;96,1669,8,0;97,1660,12,0;98,1658,13,0;99,0,0,8002;100,1658,17,0;101,
1637,6,0;102,1436,9,0;103,1414,12,0;104,0,0,8035;105,1413,18,0;106,1429,32,0;107,0,0,8
035;108,1574,6,0;109,1557,14,0;110,1558,12,0;111,1573,7,0;112,0,0,8035;113,0,0,8035;11
4,0,0,8035;115,0,0,8035;116,0,0,8035;117,0,0,8035;118,0,0,8035;119,2285,16,0;120,0,0,8
002;121,2156,44,0;122,2083,61,0;123,1831,49,0;124,1978,70,0;125,1938,64,0;126,1890,93,
0;127,1851,96,0;128,1810,99,0;129,1772,107,0;130,1736,118,0;131,1704,125,0;132,1670,10
7,0;133,1641,130,0;134,1613,131,0;135,1585,153,0;136,1560,122,0;137,1537,168,0;138,151
4,169,0;139,1490,157,0;140,1469,175,0;141,1452,195,0;142,1432,103,0;143,0,0,8035;144,9
33,237,0;145,824,21,0;146,0,21,8021;147,974,22,0;148,0,0,8035;149,1071,47,0;150,1057,2
56,0;151,0,0,8035;152,884,201,0;153,892,41,0;154,910,33,0;155,935,21,0;156,965,10,0;15
7,998,9,0;158,1167,15,0;159,1223,260,0;160,1209,124,0;161,1207,273,0;162,1195,243,0;16
3,1172,101,0;164,0,0,8035;165,0,0,8035;166,0,0,8035;167,0,0,8035;168,0,0,8035;169,0,0,
8035;170,0,0,8035;171,0,0,8035;172,1180,79,0;173,1164,222,0;174,1154,298,0;175,1153,97
,0;176,1148,314,0;177,1147,311,0;178,1096,48,0;179,1147,306,0;180,1146,317,0;181,1147,
312,0;182,1148,323,0;183,1150,318,0;184,1151,311,0;185,1153,319,0;186,1154,321,0;187,1
157,321,0;188,1159,307,0;189,1162,307,0;190,1165,317,0;191,1169,299,0;192,1174,283,0;1
93,1179,299,0;194,1184,295,0;195,1192,251,0;196,1199,197,0;197,1208,138,0;198,1217,78,
0;199,1223,25,0;200,0,0,8035;201,0,0,8035;202,0,0,8035;203,0,0,8035;204,1195,12,0;205,
1179,82,0;206,1191,147,0;207,1207,210,0;208,0,0,8035;209,1255,127,0;210,1264,283,0;211
,1348,200,0;212,1363,227,0;213,1379,210,0;214,1395,192,0;215,1413,197,0;216,1428,182,0
;217,1450,198,0;218,1470,164,0;219,1492,168,0;220,1516,156,0;221,1539,162,0;222,1563,1
41,0;223,1590,151,0;224,1616,142,0;225,1648,129,0;226,1679,123,0;227,1488,11,0;228,145
9,16,0;229,1436,41,0;230,1407,44,0;231,1392,42,0;232,1378,27,0;233,1361,34,0;234,1361,
74,0;235,1344,129,0;236,1331,124,0;237,1316,123,0;238,1305,140,0;239,1293,146,0;240,0,
0,8002;241,1271,149,0;242,0,149,8021;243,1251,161,0;244,1243,151,0;245,1235,147,0;246,
1225,171,0;247,1216,185,0;248,1202,134,0;249,1176,49,0;250,0,0,8035;251,0,0,8035;252,0
,0,8035;253,0,0,8035;254,0,0,8035;255,0,0,8035;256,0,0,8002;257,0,0,8035;258,1181,40,0
;259,1163,134,0;260,1152,216,0;261,1330,38,0;262,0,0,8035;263,0,0,8035;264,0,0,8035;26
5,0,0,8035;266,1072,315,0;267,1068,198,0;268,1067,363,0;269,1065,357,0;270,1064,385,0;
271,1065,363,0;272,1066,373,0;273,1068,366,0;274,1070,367,0;275,1070,360,0;276,1073,35
9,0;277,1075,365,0;278,1078,344,0;279,1081,341,0;280,1086,43,0;281,0,0,8035;282,0,0,80
35;283,16972,24,0;284,16962,46,0;285,16951,22,0;286,0,0,8035;287,0,0,8035;288,0,0,8035
;289,0,0,8035;290,510,67,0;291,516,117,0;292,520,155,0;293,524,195,0;294,524,216,0;295
,525,214,0;296,526,243,0;297,530,192,0;298,0,0,8035;299,1201,33,0;300,1167,42,0;301,11
39,130,0;302,1111,21,0;303,1091,21,0;304,1077,27,0;305,0,0,8035;306,0,0,8035;307,2092,
10,0;308,0,0,8035;309,1930,12,0;310,1921,6,0;311,0,0,8035;312,1273,219,0;313,1663,17,0

```
        ;333,3098,25,0;334,0,0,8035;335,2977,15,0;336,0,0,8035;337,1425,40,0;338,1392,11,0;339
        ,1388,23,0;340,1379,112,0;341,1383,147,0;342,1394,183,0;343,1413,198,0;344,0,0,8002;34
        5,2854,58,0;346,2843,64,0;347,2831,52,0;348,2822,61,0;349,2811,69,0;350,2814,70,0;351,
        0,0,8035;352,0,0,8035;353,0,0,8035;354,0,0,8035;355,0,0,8035;356,0,0,8035;357,0,0,8035
        ;358,0,0,8035;359,2816,8,0;ROTATION_SPEED,5.08;'
        sendMessage(conn,msg) # se envia al cliente web

        rec = None
        while rec == None:
            rec = get_Data(conn);
            if rec == None:
                print 'No data: ' + rec

        print 'Data: ' + rec

    print 'Cerrando la conexion'
    ws_close(conn)
    return False

################################################################################
################################################################################
################################################################################
#================================== MAIN ================================================
if __name__ == '__main__':
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    print 'Socket created'

    #Bind socket to local host and port
    try:
        s.bind((HOST, PORT))
    except socket.error as msg:
        print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + str(msg[1])
        sys.exit()

    print 'Socket bind complete'
    loop = True
    q_stop.put("first") # Se pone dato para saltar el bucle de espera inicial

    #Start listening on socket
    s.listen(1)
    print 'Socket now listening'

    # Bucle principal, escucha conexiones entrantes y genera threads cuando hay conexion
    while loop:

        # Espera a que se elimine la conexion existente
        while q_stop.empty():
            pass

        # Ha llegado instruccion a "q_stop"
        dat = q_stop.get()
        # Revisa si hay que seguir aceptando conexiones o se sale
        if dat == 'Stop': # Parar el programa
            print 'Main Stop'
            loop = False
            s.close()
```

```
            time.sleep(2)
            sys.stdout.flush()
            sys.exit()
            break
        if data == 'Close':
            data = ''
            break

        print '\nWaiting for a connection...\n'

        # Espera a establecer conexion - llamada bloqueante
        conn, addr = s.accept()
        #start = time.time() # Toma el tiempo cuando se realiza la conexion para calcular el
        timestamp
        print 'Connected with ' + addr[0] + ':' + str(addr[1])
        print 'WS_Handshaking: '+ str(do_handshaking(conn))

        #loop = testing_conn(conn) # Para testear la conexion (para usar comentar de aqui al
        final)

        #Enviar mensaje sobre la deteccion del puerto serie (needed, handshaking)
        device = get_serial_port() # Port type detect

        run = False
        if device != 'null':
            run = True
            sendMessage(conn,'Serial OK') #+chr(10)
            print 'Serial OK'
        else:
            sendMessage(conn,'Serial KO') #+chr(10)
            'Serial KO'
            ws_close(conn)
            s.close()
            time.sleep(2)
            sys.stdout.flush()


        if run:
            # Esperar mensaje de modo de funcionamiento (DataMode [listener,constant,moody])
            (needed, handshaking)
            repeat = True
            while repeat:
                dat = get_Data(conn)
                print dat
                dat = dat.replace(chr(26),"") #remove SUB
                dat = dat.replace(chr(10),"") #remove NL
                mod = dat.split()
                if (mod[0] == 'DataMode') and ( mod[1] == 'listener' or mod[1] ==
                'constant'or mod[1] == 'moody'):
                    repeat = False
                    sendMessage(conn,mod[0] + ' '+ mod[1]+ ' OK') #+chr(10)
                    mode = mod[1]

            # Se generan nuevos threads, primer argumento, nombre de la func a ejecutar
            # segundo, tupla con los argumentos de la funcion
            # Se diferencia entre windows y linux, en windows threads virtuales, en windows
            multihreading real
```

```
            if sys.platform == 'win32':
            #### Threads virtuales #####
                # Thread que escucha el socket y encola los mensajes entrantes desde la
                aplicacion
                if mode == 'listener':
                    start_new_thread(serial_thread_listener ,(q_cli2neato,q_neato2cli,q_stop))
                elif mode == 'constant':
                    start_new_thread(serial_thread_constant ,(q_cli2neato,q_neato2cli,q_stop))
                else: #mode == moody
                    start_new_thread(serial_thread_moody ,(q_cli2neato,q_neato2cli,q_stop))
                time.sleep(3.5) # Esperar a que el thread serie se estabilice
                start_new_thread(sckt_lstn_thread ,(conn,q_cli2neato))
                start_new_thread(sckt_wrt_thread ,(conn,q_neato2cli))
            else:
            #### Multithreading #####
                # Thread que escucha el socket y encola los mensajes entrantes desde la
                aplicacion
                if mode == 'listener':
                    ser = Process(target=serial_thread_listener,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
                elif mode == 'constant':
                    ser = Process(target=serial_thread_constant,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
                else: #mode == moody
                    ser = Process(target=serial_thread_moody,
                    args=(q_cli2neato,q_neato2cli,q_stop))
                    ser.start()
                time.sleep(3.5)  # Esperar a que el thread serie se estabilice
                sckt_lstn = Process(target=sckt_lstn_thread, args=(conn,q_cli2neato))
                sckt_wrt = Process(target=sckt_wrt_thread, args=(conn,q_neato2cli))
                sckt_lstn.start()
                sckt_wrt.start()

                ser.join()
                sckt_lstn.join()
                sckt_wrt.join()

    s.close()
    sys.exit()
```

Anexo D

Código interfaz Matlab

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\gui\NeatoInterface.m jueves, 22 de octubre de 2015 3:27

```
function varargout = NeatoInterface(varargin)
% NEATOINTERFACE MATLAB code for NeatoInterface.fig
%      NEATOINTERFACE, by itself, creates a new NEATOINTERFACE or raises the existing
%      singleton*.
%
%      H = NEATOINTERFACE returns the handle to a new NEATOINTERFACE or the handle to
%      the existing singleton*.
%
%      NEATOINTERFACE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in NEATOINTERFACE.M with the given input arguments.
%
%      NEATOINTERFACE('Property','Value',...) creates a new NEATOINTERFACE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before NeatoInterface_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to NeatoInterface_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help NeatoInterface

% Last Modified by GUIDE v2.5 23-Apr-2015 15:07:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @NeatoInterface_OpeningFcn, ...
    'gui_OutputFcn',  @NeatoInterface_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

global radio;
global backColor;
global borderColor;
global axisColor;
global axisColorb;
global axisColorm;
global p_color;
global l_color;
global path;
global m_ejec;
```

```
radio = 5;
m_ejec = 4;
backColor = [0 0 0];
borderColor = [0.4 0.4 0.4];
axisColor = [0.6 0.6 0.6];
axisColorb = [0.2 0.2 0.2];
axisColorm = [0.8 0.8 0.8];
p_color = [52 255 30]/255; % Color de punto radar
l_color = [20 78 14]/255; % Color de linea radar;
path = '';


% --- Executes just before NeatoInterface is made visible.
function NeatoInterface_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to NeatoInterface (see VARARGIN)

% Choose default command line output for NeatoInterface
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes NeatoInterface wait for user response (see UIRESUME)
% uiwait(handles.figure1);
%Inicializacion de las graficas
radar_init();
map_init();



% --- Outputs from this function are returned to the command line.
function varargout = NeatoInterface_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider_i_Callback(hObject, eventdata, handles)
global mx;
% hObject    handle to slider_i (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% Actualizar valor del label
lbl = handle(findobj('Tag','lbl_index')); %Label index handle
i = round(get(hObject,'Value'));
```

```
set(lbl,'String',i);
updt_grphs(i);



% --- Executes during object creation, after setting all properties.
function slider_i_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_i (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in btn_open.
function btn_open_Callback(hObject, eventdata, handles)
% hObject    handle to btn_open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global path;
global data_enc;
global lds_dis;
global lds_err;
global lndmrk;
global trajec;
global pk;
global mx;
global my;
global pw;
global eh;

eh = [];

global chk_traj;
global chk_lray;
global chk_landmarks;
global chk_hold;
global chk_elips;
% Inicializacion de check_boxes
hndlc = handle(findobj('Tag','chk_lray')); %chk_lray handle;
chk_lray  = get(hndlc,'Value');

hndlc = handle(findobj('Tag','chk_hold')); %chk_hold handle;
chk_hold = get(hndlc,'Value');

%Se obiene el path a leer
path = uigetdir;
%display(path); %C:\Users\RJ\Dropbox\UPC\PFC\PROYECTO\Matlab\data_cuadrit
% Cargar datos de los archivos (laser y ruedas) lds_dis, lds_err, data_enc
%data_enc = importfile_enc(strcat(path,'\','data_enc.txt'));
%lds_dis = importfile_lds_ed(strcat(path,filesep,'lds_dis.txt'));
%lds_err = importfile_lds_ed(strcat(path,'\','lds_err.txt'));
if path ~= 0

    width = 243; % wheel axes distance [mm]
```

```
    assignin('base', 'width', width);

    r_wheel = 38.5; % wheel radius [mm]
    assignin('base', 'r_wheel', r_wheel);


    % Lee datos de laser
    var = matfile(strcat(path,filesep,'lds_dis.mat')); % Laser data
    lds_dis = var.flds_dism;
    assignin('base', 'lds_dis', lds_dis);

    % Lee datos de encoders
    var = matfile(strcat(path,filesep,'data_enc.mat')); % Encoders data
    data_enc = var.fencm;
    assignin('base', 'data_enc', data_enc);

    % Lee landmarks
    hndlc = handle(findobj('Tag','chk_landmrk')); %chk_landmarks handle;
    if exist(strcat(path,filesep,'data_landmark.txt'), 'file') == 2 % Trajectory file exists
        lndmrk = importfile_landmrk(strcat(path,filesep,'data_landmark.txt'),1,4);
        assignin('base', 'lndmrk', lndmrk);
        set(hndlc,'Value',1);
        chk_landmarks = get(hndlc,'Value');
    else
        set(hndlc,'Value',0);
        chk_landmarks = get(hndlc,'Value');
    end

    % Lee ellipse de error
    %if exist('pk','var') && ~isempty(pk) % Existe en workspace
    %    pk = getVariable('base','pk');
    %else
    hndlc = handle(findobj('Tag','chk_elips')); %chk_elips handle;
    if exist(strcat(path,filesep,'pk.mat'), 'file') == 2 % Error file exists
        var = matfile(strcat(path,filesep,'pk.mat'));
        pk = var.Pk;
        assignin('base', 'pk', pk);
        set(hndlc,'Value',1);
        chk_elips = get(hndlc,'Value');
    else
        set(hndlc,'Value',0);
        chk_elips = get(hndlc,'Value');
    end
    %end

    % Calcular coordenadas laser
    if exist(strcat(path,filesep,'lds_xy.mat'), 'file') == 2
        var = matfile(strcat(path,filesep,'lds_xy.mat'));
        mx = var.mx;
        my = var.my;
        assignin('base', 'ldx', mx);
        assignin('base', 'ldy', my);
    else
        coords(lds_dis); % Calcular coordenadas
        lds_xy = strcat(path,filesep,'lds_xy.mat');
        save(lds_xy,'mx','my');
    end
```

```
    %######## Solo en el caso que exista #########

    %Calcular trayectoria y puntos laser mundo
    hndlc = handle(findobj('Tag','chk_traj')); %chk_traj handle;
    if exist(strcat(path,filesep,'data_traj.mat'), 'file') == 2
        %Trayectoria ya calculada
        var = matfile(strcat(path,filesep,'data_traj.mat'));
        trajec = var.trajec;
        assignin('base', 'trajec', trajec);
        set(hndlc,'Value',1);
        chk_traj = get(hndlc,'Value');
    else
        %Comprobar que existe el archivo de calculo de trayectoria
        cpath = pwd;
        if exist(strcat(cpath,filesep,'calc_traj.m'), 'file') == 2
            trajec = calc_traj(path,data_enc);
            calc_pw(mx,my,trajec,path);
            chk_traj = get(hndlc,'Value');
        else
            set(hndlc,'Value',0);
            chk_traj = get(hndlc,'Value');
        end
    end

    % Si existe el archivo de datos de laser en mundo, lo carga
    if exist(strcat(path,filesep,'pw.mat'), 'file') == 2 % World laser data file
        var = matfile(strcat(path,filesep,'pw.mat'));
        pw = var.pw;
        assignin('base', 'pw', pw);
    end

    display(exist('pw','var'));

    %######## FIN - Solo en el caso que exista #########

    % Set slider dimensions
    sli = handle(findobj('Tag','slider_i')); %Slider handle

    sliderMin  = 1;
    [siz,leng] = size(lds_dis);
    sliderMax = siz -1;
    sliderStep = [1, 1] / (sliderMax - sliderMin);

    set(sli, 'Min', sliderMin);
    set(sli, 'Max', sliderMax);
    set(sli, 'SliderStep', sliderStep);
    set(sli, 'Value', sliderMin); % set to beginning of sequence

    % Establece el valor del label a 1
    hndlc = handle(findobj('Tag','lbl_index')); %lbl_index handle;
    set(hndlc,'String',1);

    % Actualizar graficos
    updt_grphs(1);

end
```

```
% --- Executes on button press in btn_connect.
function btn_connect_Callback(hObject, eventdata, handles)
% hObject    handle to btn_connect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Codigo para generar los archvios despues del recorrido -- OK
%%c = clock; % Tiempo actual
%%dir = strcat( d2str(c(1)), '_' ,d2str(c(2)), '_' ,d2str(c(3)) ,'-', d2str(c(4)), '_'
,d2str(c(5)) );
%%mkdir(dir); % Se genera la carpeta con el tiempo actual

%%hist = [1 2 3 4]; % Datos random
%%nfilem = strcat(dir,'/','data.mat'); % Se genera la ruta al archivo
%%save(nfilem,'hist'); % Se guarda la variable en el archivo

%--------------------------------------------------------------------------
% Ocultar boton connect
set(hObject,'Visible','off');
% Mostrar boton Stop
hbstop = handle(findobj('Tag','btn_Stop'));
set(hbstop,'Visible','on');
% Pasar parámetros, handler ventana y handler botones
connect(hObject,hbstop);


% --- Executes on button press in btn_Stop.
function btn_Stop_Callback(hObject, eventdata, handles)
% hObject    handle to btn_Stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Ocultar boton Stop
set(hObject,'Visible','off');
% Mostrar boton connect
hbconn = handle(findobj('Tag','btn_connect'));
set(hbconn,'Visible','on');

% --- Executes on button press in chk_traj.btn_connect
function chk_traj_Callback(hObject, eventdata, handles)
% hObject    handle to chk_traj (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chk_traj
global chk_traj;
global trajec;
global pw;

hndlc = handle(findobj('Tag','chk_traj')); %chk_traj handle;
chk_traj = get(hndlc,'Value');

if chk_traj
    if evalin('base','exist(''trajec'',''var'')')
        trajec = evalin('base','trajec');
    end
    if evalin('base','exist(''pw'',''var'')')
```

```
        pw = evalin('base','pw');
    end
end

hndlc = handle(findobj('Tag','lbl_index')); %Slider handle;
index = get(hndlc,'String');
updt_grphs(round(str2num(index)));


% --- Executes on button press in chk_landmrk.
function chk_landmrk_Callback(hObject, eventdata, handles)
% hObject    handle to chk_landmrk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chk_landmrk
global chk_landmarks;
global lndmrk;
global pw;
hndlc = handle(findobj('Tag','chk_landmrk')); %chk_landmarks handle;
chk_landmarks = get(hndlc,'Value');

if chk_landmarks
    if evalin('base','exist(''pw'',''var'')')
        pw = evalin('base','pw');
    end
end

hndlc = handle(findobj('Tag','lbl_index')); %lbl_index handle;
index = get(hndlc,'String');
updt_grphs(round(str2num(index)));


% --- Executes on button press in chk_lray.
function chk_lray_Callback(hObject, eventdata, handles)
% hObject    handle to chk_lray (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chk_lray
global chk_lray;
hndlc = handle(findobj('Tag','chk_lray')); %chk_lray handle;
chk_lray  = get(hndlc,'Value');

hndlc = handle(findobj('Tag','lbl_index')); %Slider handle;
index = get(hndlc,'String');
updt_grphs(round(str2num(index)));


% --- Executes on button press in chk_hold.
function chk_hold_Callback(hObject, eventdata, handles)
% hObject    handle to chk_hold (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chk_hold
global chk_hold;
```

```
hndlc = handle(findobj('Tag','chk_hold')); %chk_hold handle;
chk_hold = get(hndlc,'Value');

hndlc = handle(findobj('Tag','lbl_index')); %Slider handle;
index = get(hndlc,'String');
updt_grphs(round(str2num(index)));


% --- Executes on button press in chk_elips.
function chk_elips_Callback(hObject, eventdata, handles)
% hObject    handle to chk_elips (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chk_elips
global chk_elips;
global pk;

hndlc = handle(findobj('Tag','chk_elips')); %chk_hold handle;
chk_elips = get(hndlc,'Value');

if chk_elips
    if evalin('base','exist(''pk'',''var'')')
        pk = evalin('base','pk');
    end
end

hndlc = handle(findobj('Tag','lbl_index')); %Slider handle;
index = get(hndlc,'String');
updt_grphs(round(str2num(index)));


% =============== FUNCIONES ==========================

% Calcula las coordenadas de los puntos
function coords(info)
global mx;
global my;

mx = [];
my = [];

wt = waitbar(0,'Calculating coords...');

[siz,leng] = size(info);
for i=1:siz % Recorre la matriz
    d = info(i,:);

    x = [];
    y = [];

    for k=2:length(d)
        xx = (cosd(k-2) * d(k)) / 1000;
        yy = (sind(k-2) * d(k)) / 1000;

        if (xx < 6.1) && (yy < 6.1) && (xx > -6.1) && (yy > -6.1) && (xx ~= 0 && yy ~= 0)
            %if (xx < 6.1) && (yy < 6.1) && (xx > -6.1) && (yy > -6.1) && (xx ~= 0 && yy ~= 0)
```

```
            x = [x xx];
            y = [y yy];
        else
            x = [x NaN];
            y = [y NaN];
        end
    end

    waitbar(i/siz,wt,'Calculating coords...');

    mx=[mx;x];
    my=[my;y];
end

close(wt);

assignin('base', 'ldx', mx);
assignin('base', 'ldy', my);

%Establece el look & feel del mapa 2D
function map_init()
global axisColorm;
global axisColorb;
global m_ejec;

map = findobj('Type','axes','Tag','axes2');
axes(map);

d = m_ejec;
xlim([-d-0.1 d+0.1]); ylim([-d-0.1 d+0.1]);
hold on;
%view(90,90);

r = d;
%draw strong axis lines
for x = -r: 1: r
    line([x x],[-r r], ... % [x1 x2] [y1 y2]
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColorb, ...
        'parent', map);

    line([-r r],[x x], ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColorb, ...
        'parent', map);
end

r = d+0.5;
for x = -r: 1: r
    line([x x],[-r r], ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColorm, ...
        'parent', map);
```

```
    line([-r r],[x x], ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColorm, ...
        'parent', map);
end

% Para que se vea cuadrado
axis square;

%EStablece el look & feel del radar
function radar_init()
global radio;
global backColor;
global borderColor;
global axisColor;
global axisColorb;

% Antialiasing mode with OpenGL (all the graphics are nicer)
win = gcf;
set(win, 'renderer', 'opengl');
set(win,'DefaultLineLineSmoothing','on');
set(win,'DefaultPatchLineSmoothing','on');

% Especifica el axes1 como destino del radar
radarAxes = findobj('Type','axes','Tag','axes1');
axes(radarAxes);

%%%%%% Radar graph init %%%%%%
r = radio;
xlim([-r-0.1 r+0.1]); ylim([-r-0.1 r+0.1]);
hold on;
axis off;
view(-90,90);

th = 0:pi/40:2*pi;
x = r * cos(th);
y = r * sin(th);
patch('parent', radarAxes, ...
    'XData', [x x(1)], ...
    'YData', [y y(1)], ...
    'FaceColor', backColor, ...
    'EdgeColor', borderColor, ...
    'LineWidth', 2, ...
    'HitTest', 'off');

% draw axis radius 1m
for i = 1:r-1
    th = 0:pi/50:2*pi;
    x = i * cos(th);
    y = i * sin(th);
    line(x, y, ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColor, ...
        'parent', radarAxes);
end
```

```
% draw axis radius 0.5m
for i = 0.5:1:r-0.5
    th = 0:pi/50:2*pi;
    x = i * cos(th);
    y = i * sin(th);
    line(x, y, ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColorb, ...
        'parent', radarAxes);
end

% draw orientation mark
%set(this.gh.ring, 'Parent', this.hAxes);
%th = 0:pi/5:2*pi;
%x = 0.05 * cos(th) + r;
%y = 0.05 * sin(th);
%patch('parent', hgtransform(), ...
%    'XData', x, ...
%    'YData', y, ...
%    'FaceColor', axisColor, ...
%    'EdgeColor', axisColor, ...
%    'LineWidth', 2, ...
%    'HitTest', 'off');

% draw axis lines
for th = 0: pi/4: 2*pi
    x = r * cos(th);
    y = r * sin(th);
    line([0 x],[0 y], ...
        'LineStyle', '-.', ...
        'LineWidth', 0.1, ...
        'Color', axisColor, ...
        'parent', radarAxes);
end

% Para que se vea cuadrado
axis square;


%Dibuja punto con las caracteristicas indicadas
function drawPoints(axe,tag, p, s, color, marker)  %win.drawPoints('p', points , 40, [52 255
30]/255, 'x'); %[36 230 15]
if isempty(p) || numel(p) == 0
    p = [NaN; NaN];
end
h = handle(findobj(axe, 'Tag', tag));
if isempty(h)
    h = handle(scatter( p(1,:), p(2,:),s,color,marker));
    h.Tag = tag;
    h.Parent = axe;
else
    h.XData = p(1,:);
    h.YData = p(2,:);
    h.CData = color;
    h.SizeData = s;
```

```
    h.marker = marker;
end

%Dibuja linea con las caracteristicas indicadas
function drawRays(axe,tag, p, color, width, style) %win.drawRays('r', points , [20 78
14]/255, 1 ,'-'); %[208 247 239]
if isempty(p) || numel(p) == 0
    p = [NaN; NaN];
end
X = repmat([0 0 NaN], 1, size(p,2));
Y = repmat([0 0 NaN], 1, size(p,2));
X(2:3:end) = p(1,1:end);
Y(2:3:end) = p(2,1:end);
h = findobj(axe, 'Tag', tag);
if isempty(h)
    h = line(X, Y, 'Color', color, 'LineStyle', style, 'LineWidth', width);
    set(h, 'Tag', tag);
    set(h, 'Parent', axe);
else
    set(h, 'XData', X, 'YData', Y, 'Color', color, 'LineStyle', style, 'LineWidth', width);
end

%Dibuja punto con las caracteristicas indicadas
function drawPointsM(axe,tag, p, s, color, marker)  %win.drawPoints('p', points , 40, [52
255 30]/255, 'x'); %[36 230 15]
if isempty(p) || numel(p) == 0
    p = [NaN; NaN];
end
h = findobj(axe, 'Tag', tag);
if isempty(h)
    %h = plot( p(1,:), p(2,:),s,color,marker);
    h = plot(p(1,:), p(2,:), '.r', 'MarkerSize',s, 'Color', color);
    set(h, 'Tag', tag);
    set(h, 'Parent', axe);
else
    set(h, 'XData', p(1,:),'YData', p(2,:));
    set(h, 'Color', color);
    set(h, 'MarkerSize', s);
end


%Dibuja una elipse
function drawEllipse(axe,ell_data,pos,color)
global eh;
if isempty(pos) || numel(pos) == 0
    pos = [NaN; NaN];
end
if isempty(eh)
    eh = plot_ellipse(ell_data,pos,color);
else
    plot_ellipse(ell_data,pos,color,'alter',eh);
end

%Dibuja la trayectoria
function drawTraj(axe,tag, x,y)
h = findobj(axe, 'Tag', tag);
if isempty(h)
```

```
    h = plot(axe,x,y);
    set(h, 'Tag', tag);
    set(h, 'Parent', axe);
else
    set(h, 'XData', x,'YData',y);
end

%Dibuja los landmarks
function drawLndmark(axe,tag, x,y)
h = findobj(axe, 'Tag', tag);
if isempty(h)
    h = plot(axe, x, y, '.r', 'MarkerSize',30, 'Color', [19 205 32]/255);
    set(h, 'Tag', tag);
    set(h, 'Parent', axe);
else
    set(h, 'XData', x,'YData',y);
end


% Actualiza las graficas
function updt_grphs(i)
global mx;
global my;
global p_color;
global l_color;
global trajec;
global lndmrk;
global pk;
global pw;
global chk_traj;
global chk_lray;
global chk_landmarks;
global chk_elips;
global chk_hold;
global eh;

%filtrado de puntos
x = mx(i,:);
y = my(i,:);
[s,l] = size(x);
u=1;
while u<l+1
    if(x(u) == 0 && y(u)==0)
        x(u) = [];
        y(u) = [];
    end
    [s,l] = size(x);
    u = u + 1;
end

% --------------- RADAR MAP --------------------
radarAxes = findobj('Type','axes','Tag','axes1');
drawPoints(radarAxes,'p', [x; y]  , 40, p_color, 'x'); %[36 230 15]
if chk_lray == 1
    drawRays(radarAxes,'r', [x; y] , l_color , 1 ,'-');
else
    ht = findobj(radarAxes, 'Tag', 'r');
```

```
    delete(ht);
end
drawPoints(radarAxes,'c', [0; 0]  , 40, [255 43 245]/255, 'x');

% ##################### 2D MAP ####################
%display(trajec);
if length(trajec) ~= 0 % Trajectory file exists

    %--------------- TRAJECTORY -----------------
    mapAxes = findobj('Type','axes','Tag','axes2');
    if chk_traj == 1
        drawTraj(mapAxes,'traj',trajec(:,1),trajec(:,2));
    else
        ht = findobj(mapAxes, 'Tag', 'traj');
        delete(ht);
    end
    %---------------- ROBOT ------------------------
    p = trajec(i,:);
    drawPointsM(mapAxes,'rob', [p(1);p(2)] , 30, [134 50 33]/255, 'o'); %[193 14 50]/255
end

% ------------- REAL LANDMARKS ------------------------
if length(lndmrk) ~= 0
    mapAxes = findobj('Type','axes','Tag','axes2');
    if chk_landmarks == 1
        drawLndmark(mapAxes,'lndmrks2',lndmrk(:,1), lndmrk(:,2));
    else
        ht = findobj(mapAxes, 'Tag', 'lndmrks2');
        delete(ht);
    end
end

% ------------- ERROR ELLIPSES ------------------------
if ~isempty(pk)
    mapAxes = findobj('Type','axes','Tag','axes2');
    if chk_elips == 1
        p = trajec(i,:);
        drawEllipse(mapAxes,pk.signals.values(1:2,1:2,i),[p(1);p(2)],'g');
    else
        if ~isempty(eh)
            delete(eh);
            eh = [];
        end
    end
end

% ------------- LANDMARKS FROM ROBOT -------------

[s,l] =size(pw);
if s > 0
    if chk_hold == 1
        if i < 62
            x = pw(1,:,1:i);
            y = pw(2,:,1:i);
        else
            x = pw(1,:,i-60:i);
            y = pw(2,:,i-60:i);
```

```
        end

        x = x(:)';
        y = y(:)';
        %x(isnan(x)) = [];
        %y(isnan(y)) = [];
        %scatter(x,y);
        drawPoints(mapAxes,'lndmrksr', [x;y] , 30, [234 50 33]/255, 'o');
    else
        x = pw(1,:,i);
        x = x(:)';
        y = pw(2,:,i);
        y = y(:)';
        %x(isnan(x)) = [];
        %y(isnan(y)) = [];
        %scatter(x,y);
        drawPoints(mapAxes,'lndmrksr', [x;y] , 30, [234 50 33]/255, 'o');
    end
end
```

```
classdef socket
    properties
        skt
        input
        output
    end
    methods(Static)
        %Contructora (conecta directamente con el servidor)
        function obj = socket(ip,port)
            if nargin > 0
                try
                    obj.skt = java.net.Socket(ip,port);
                    obj.input =
                    java.io.BufferedReader(java.io.InputStreamReader(obj.skt.getInputStream())
                    ,12288);
                    obj.output =
                    java.io.PrintWriter(obj.skt.getOutputStream());
                catch e
                    display('Connection not established, check correct ip:port, and server
                    on');
                    obj.skt = 'null';
                end
            end
        end

        %Sends message and waits for response
        function txt = sendMsg(obj,msg)
            obj.output.println(msg);
            obj.output.flush();
            txt = obj.input.readLine();
        end

        %Waits for a message
        function txt = getMsg(obj)
            txt = obj.input.readLine();
        end

        %Closes the current socket connection letting the server get more
        %connections
        function close(obj)
            %obj.output.println('Stop'); % Do not use, only for stopping the socket server
            service
            obj.output.println('Close');
            obj.output.flush();
        end
    end
    methods
        function sckt = get.skt(obj)
         sckt = obj.skt;
        end
    end
end
```

```
function connect(hbconnect,hbstop)

hist = {};
%--------------------------------------------------------------------
% Robot parameters
w = 243; % wheel axes distance [mm]
%r_wheel = 3.85; % wheel radius [cm]
%--------------------------------------------------------------------
% User interface parameters
alpha = 0; %  incremental angular rotation while moving [rad]
speed_factor = 0.5; % speed security factor
speed = 0; % current speed [mm/s] (max 300)
% Key arrows increment (joystick) speed and turn
inc_speed = 30; % a tenth of the max linear speed [mm/s] (up/down arrows)

theta_dot = 0;  % current robot angular speed [rad/seg]
max_theta_dot = 2.51; % [rad/s] 2 pi / time needed for a turn (perimeter / max linear speed)
inc_ang_turn = 0.251 ;  % a tenth of the max angular speed [rad/s] (left/right arrows)
time_for_a_turn = (2 * pi * w) / (inc_speed * 10); % Robot perimeter divided by max_speed
(300) [sec] (5.0894)
%--------------------------------------------------------------------

% UI Datos de conexión con Neato
prompt = {'Connection information',};
dlg_title = 'Connection';
num_lines = 1;
def = {'192.168.1.154:20000',};
%def = {'192.168.1.12:20000',};
%def = {'147.83.173.108:20000',};
%def = {'127.0.0.1:20000',};
conn = inputdlg(prompt,dlg_title,num_lines,def);
conexion = strsplit(conn{1},':');

% Neato Connection
s = socket(conexion{1},str2double(conexion{2}));

% If no connection, no code execution
if length(strfind(class(s.skt()),'java.net.Socket')) == 1

    cont = 1;

    % Check if serial port is OK (required handshaking)
    data = s.getMsg(s);
    ok = strfind(data, 'OK');
    if length(ok) == 1
        display('Serial port (Raspberry-Neato) OK');
    else
        display('Serial port (Raspberry-Neato) not found, please check cable connection and
        Neato power on, and restart Raspberry');
        cont = 0;
        set(hbstop,'Visible','off');
        hbconn = handle(findobj('Tag','btn_connect'));
        set(hbconn,'Visible','on');
    end

    if(cont)
```

```
        % Send data mode message (required handshaking)
        data = s.sendMsg(s,'DataMode constant'); %(DataMode [listener,constant,moody])
        display(data);

        % Wait lidar start
        wait_lidar();

        % Crea la funcion listener del pulsado de los botones en la UI
        rax = findobj('Type','axes','Tag','axes1');
        uiid = get(rax,'Parent');

        set(uiid,'KeyPressFcn',{@KeyPressed, s});

        % Esperar a que se pulse el boton Stop
        loop = strcmp(get(hbstop,'Visible'),'on');
        while loop
            UpdDispl(s,rax);
            loop = strcmp(get(hbstop,'Visible'),'on');
            pause(0.01);
        end

        hp = handle(findobj(rax, 'Tag', 'p'));
        hr = handle(findobj(rax, 'Tag', 'r'));
        delete(hp);
        delete(hr);

        % Se cierra la conexión con Neato
        %s.sendMsg(s,'Stop');
        s.sendMsg(s,'Close');

        %se guarda el log en tipo .mat y .txt en carpeta llamada con la fecha
        c = clock;
        datime = strcat( d2str(c(1)),'_',d2str(c(2)),'_',d2str(c(3)),'-',
        d2str(c(4)),'_',d2str(c(5)));
        mkdir(datime);

        assignin('base','hist',hist);
        nfilem = strcat(datime,filesep,'data.mat');
        save(nfilem,'hist');

        nfile = strcat(datime,filesep,'data.txt');
        tofile(hist,nfile);

        %Guardar los datos en un formato mas usable (hist en data.mat)
        gen_files(nfile,datime);

    end;

    clear;
end

% ==================== FUNCIONES ====================================

%convierte formato de fecha y hora en string
    function s = d2str(v)
        s = num2str(datenum(v));
    end
```

```
% Gestiona las teclas pulsadas en el 'figure'
    function KeyPressed(obj,event,sck)
        switch event.Key
            %case 'end'
            %disp(event.Key)
            case 'uparrow'
                if speed < 300
                    speed = speed + (inc_speed * speed_factor);
                else
                    speed = 300;
                end
                [l,r] = turn(speed,theta_dot);
                msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) , '
                Speed ', num2str(abs(speed))];
                sck.sendMsg(sck,msg);
            case 'downarrow'
                if speed > -300
                    speed = speed - (inc_speed * speed_factor);
                else
                    speed = -300;
                end
                [l,r] = turn(speed,theta_dot);
                msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) , '
                Speed ', num2str(abs(speed))];
                sck.sendMsg(sck,msg);
            case 'leftarrow'
                theta_dot = theta_dot + (inc_ang_turn * speed_factor);
                if theta_dot > max_theta_dot
                    theta_dot = max_theta_dot;
                end
                [l,r] = turn(speed,theta_dot); % Concatenacion de str de l y r
                if speed == 0
                    msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) ,
                    ' Speed ', num2str(abs(r/time_for_a_turn))];
                else
                    msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) ,
                    ' Speed ', num2str(abs(speed))];
                end
                sck.sendMsg(sck,msg);
            case 'rightarrow'
                theta_dot = theta_dot - (inc_ang_turn * speed_factor);
                if theta_dot < -max_theta_dot
                    theta_dot = -max_theta_dot;
                end
                [l,r] = turn(speed,theta_dot);
                if speed == 0
                    msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) ,
                    ' Speed ', num2str(abs(r/time_for_a_turn))];
                else
                    msg = ['SetMotor LWheelDist ', num2str(l) ,' RWheelDist ', num2str(r) ,
                    ' Speed ', num2str(abs(speed))];
                end
                sck.sendMsg(sck,msg);
            case 'space'
                msg = 'SetMotor RWheelDisable LWheelDisable';
                sck.sendMsg(sck,msg);
```

```
                msg = 'SetMotor RWheelEnable LWheelEnable';
                sck.sendMsg(sck,msg);

                %                 ht = timerfind('Tag','reptimer');
                %                 if ~isempty(ht)
                %                     delete(ht);
                %                 end

                theta_dot = 0;
                speed = 0;
            otherwise
                disp(event.Key)
        end
    end

% Calcula las distancias por rueda
    function [l,r] = turn(speed,theta_dot)
        arc_c = speed * time_for_a_turn; % Robot center arc displacement of robot center
        alpha = theta_dot * time_for_a_turn;  % Incremental angular rotation while moving
        [rad]

        r = arc_c + (w * alpha);
        l = arc_c - (w * alpha);
    end


% This function is called by the timer to display one frame of the figure
    function UpdDispl(skt,raxes)
        [x,y] = get_data(skt);
        points = [x;y];
        [r,t] = size(points);

        if(r > 1 && t > 1)
            drawPointsc(raxes,'p', points , 40, [52 255 30]/255, 'x'); %[36 230 15]
            drawRaysc(raxes,'r', points , [20 78 14]/255, 1 ,'-'); %[208 247 239]
        end
    end

    function hst = record(hst,dat)
        [o,p] = size(hst);
        hst{1,p+1} = dat;
    end

    function tofile(dt,ff)
        % open a file for writing
        fid = fopen(ff, 'w');
        for k=1:length(dt)
            fprintf(fid, '%s  %s  %s  %s  %s \n', strrep(dt{k},char(26),'#')); % Se
            reemplaza el 'SUB' por #
        end
        fclose(fid);
    end


    function [x,y] = get_data(skt)
        x=[];
        y=[];
```

```
        %Va recogiendo los msjs del socket
        %tic;
        m = [char(skt.getMsg(skt)),char(10)]; %char para pasar de java.langString a string
        de matlab
        %toc
        hist = record(hist,m);
        %display(m);
        %---------
        [data,lds] = data_conv(m);
        %display(dd);
        %assignin('base','lds',lds);
        [x,y] = calc_coords(lds);
    end

    function [x,y] = calc_coords(lds)
        x = [];
        y = [];

        [l,sz] = size(lds);

        if sz > 350
            i = 3;
            x = [];
            y = [];
            while i <= (sz-1)
                v = str2num(lds{i});
                if numel(v) == 4 && v(4) == 0
                    xx = (cosd(v(1)) * v(2)) / 1000;
                    yy = (sind(v(1)) * v(2)) / 1000;

                    if (xx < 6.1) && (yy < 6.1) && (xx > -6.1) && (yy > -6.1)
                        x = [x xx];
                        y = [y yy];
                    end
                end
                i = i+1;
            end
        end
    end

%Dibuja punto con las caracteristicas indicadas
    function drawPointsc(axe,tag, p, s, color, marker)  %win.drawPoints('p', points , 40,
    [52 255 30]/255, 'x'); %[36 230 15]
        if isempty(p) || numel(p) == 0
            p = [NaN; NaN];
        end
        h = handle(findobj(axe, 'Tag', tag));
        if isempty(h)
            h = handle(scatter( p(1,:), p(2,:),s,color,marker));
            h.Tag = tag;
            h.Parent = axe;
        else
            h.XData = p(1,:);
            h.YData = p(2,:);
            h.CData = color;
            h.SizeData = s;
            h.marker = marker;
```

```
        end
    end

%Dibuja linea con las caracteristicas indicadas
    function drawRaysc(axe,tag, p, color, width, style) %win.drawRays('r', points , [20 78
    14]/255, 1 ,'-'); %[208 247 239]
        if isempty(p) || numel(p) == 0
            p = [NaN; NaN];
        end
        X = repmat([0 0 NaN], 1, size(p,2));
        Y = repmat([0 0 NaN], 1, size(p,2));
        X(2:3:end) = p(1,1:end);
        Y(2:3:end) = p(2,1:end);
        h = findobj(axe, 'Tag', tag);
        if isempty(h)
            h = line(X, Y, 'Color', color, 'LineStyle', style, 'LineWidth', width);
            set(h, 'Tag', tag);
            set(h, 'Parent', axe);
        else
            set(h, 'XData', X, 'YData', Y, 'Color', color, 'LineStyle', style, 'LineWidth',
            width);
        end
    end

    function wait_lidar()
        wt = waitbar(0,'Waiting Lidar Start...');
        for p=1:4
            pause(1);
            waitbar(p/4,wt,'Waiting Lidar Start......');
        end
        close(wt);
    end

end
```

```
function trajec = calc_traj(path,data_enc)

%load(strcat(path,filesep,'data_enc.mat'));
%Encoder_R_L = fencm;
Encoder_R_L = data_enc;
L_sum=Encoder_R_L(:,6)./1000;
L0=cat(1,0,L_sum);
L00=cat(1,L_sum,0);
L=L00-L0;
R_sum=Encoder_R_L(:,7)./1000;
R0=cat(1,0,R_sum);
R00=cat(1,R_sum,0);
R=R00-R0;
suma_theta=zeros(length(Encoder_R_L),1);

suma_theta(1)=0;
x_w=zeros(length(Encoder_R_L),1);
y_w=zeros(length(Encoder_R_L),1);
x_w(1)=0;
y_w(1)=0;

wt = waitbar(0,'Calculating trajectory...');
for i=1:length(Encoder_R_L)
    if R(i)==L(i);
        suma_theta(i+1)=suma_theta(i);
        x_w(i+1)=x_w(i)-L(i).*sin(suma_theta(i));
        y_w(i+1)=y_w(i)+L(i).*cos(suma_theta(i));
    else
        alfa=(R(i)-L(i))/0.243;
        C=(R(i)+L(i))/2;
        suma_theta(i+1)=mod((suma_theta(i)+alfa),2*pi);
        x_w(i+1) =x_w(i)-C.*sin(suma_theta(i));
        y_w(i+1) =y_w(i)+C.*cos(suma_theta(i));
    end
    waitbar(i/length(Encoder_R_L),wt,'Calculating trajectory...');
end

close(wt);

%This code generates the file with the data [x_w,y_w,suma_theta]
wt = waitbar(0,'Generating trajectory file...');
ftraj = fopen(strcat(path,filesep,'data_traj.txt'), 'w'); % destino encoders
trajec = [];
for i=2:length(x_w)
    traj = [num2str(x_w(i)) ,',', num2str(y_w(i)),',', num2str(suma_theta(i))];
    fprintf(ftraj, '%s \n', traj);

    trajc = [x_w(i) y_w(i) suma_theta(i)];
    trajec = [trajec ; trajc];

    waitbar(i/length(x_w),wt,'Generating trajectory file...');
end

data_traj = strcat(path,filesep,'data_traj.mat');
save(data_traj,'trajec');
assignin('base', 'trajec', trajec);
```

```
close(wt);
```

```
function gen_files(ofile,path)

%Obtener numero de lineas
ofl = fopen(strcat(ofile),'r'); %origen de datos
allText = textscan(ofl,'%s','delimiter','\n');
nlines = length(allText{1});
fclose(ofl);

ofl = fopen(strcat(ofile),'r'); %origen de datos

%display(ofile);
%display(path);
%display(nlines);

%archivos que se generarán con cada tipo de dato
fenc = fopen(strcat(path,filesep,'data_enc.txt'), 'w'); % destino encoders
facl = fopen(strcat(path,filesep,'data_acl.txt'), 'w'); % destino acel
fana = fopen(strcat(path,filesep,'data_ana.txt'), 'w'); % destino analog
fdig = fopen(strcat(path,filesep,'data_dig.txt'), 'w'); % destino digital
flds_err = fopen(strcat(path,filesep,'lds_err.txt'), 'w'); % destino err_lidar
flds_qual = fopen(strcat(path,filesep,'lds_qua.txt'), 'w'); % destino qua_lidar
flds_dis = fopen(strcat(path,filesep,'lds_dis.txt'), 'w'); % destino dis_lidar

%Se especifica el formato de los datos
fprintf(fenc, '%s \n', '#Timestamp LeftWheel_RPM, RightWheel_RPM, LeftWheel_Load%,
RightWheel_Load%, LeftWheel_PositionInMM, RightWheel_PositionInMM, LeftWheel_Speed,
RightWheel_Speed');
fprintf(facl, '%s \n', '#Timestamp PitchInDegrees, RollInDegrees XInG, YInG, ZInG, SumInG');
fprintf(fana, '%s \n', '#Timestamp WallSensorInMM, BatteryVoltageInmV, LeftDropInMM,
RightDropInMM, LeftMagSensor, RightMagSensor, UIButtonInmV, VacuumCurrentInmA,
ChargeVoltInmV, BatteryTemp0InC, BatteryTemp1InC, CurrentInmA, SideBrushCurrentInmA,
VoltageReferenceInmV, AccelXInmG, AccelYInmG, AccelZInmG');
fprintf(fdig, '%s \n', '#Timestamp SNSR_DC_JACK_CONNECT, SNSR_DUSTBIN_IS_IN,
SNSR_LEFT_WHEEL_EXTENDED, SNSR_RIGHT_WHEEL_EXTENDED, LSIDEBIT, LFRONTBIT, RSIDEBIT,
RFRONTBIT');
fprintf(flds_err, '%s \n', '#Timestamp ErrorCodeHEX (For each line, one record for each
degree 0-359 | 0-> no error, other->error)');
fprintf(flds_qual, '%s \n', '#Timestamp Intensity (For each line, one record for each degree
0-359)');
fprintf(flds_dis, '%s \n', '#Timestamp DistInMM (For each line, one record for each degree
0-359)');

%fencm = zeros(nlines,9);
%faclm = zeros(nlines,7);
%fanam = zeros(nlines,18);
%fdigm = zeros(nlines,10);
%flds_errm = zeros(nlines,361);
%flds_qualm = zeros(nlines,361);
%flds_dism = zeros(nlines,361);

fencm = [];
faclm = [];
fanam = [];
fdigm = [];
flds_errm = [];
flds_qualm = [];
flds_dism = [];
```

```
wt = waitbar(0,'Generating files...');
%se lee una linea
tline = fgets(ofl);
l=1;

%cada linea se parsea para ponerla en los archivos
while ischar(tline)
    if length(strfind(tline, 'GetMotors')) ~= 0 %Si se trata de una lectura
        spl = strsplit(tline,'#'); %char(26)

        tmsp = spl{1}(1:length(spl{1})-1); % timestamp

        %display(spl);
        %display(length(spl));

        if length(spl) == 6
            % lectura encoders 10
            enc_s = strsplit(spl{2},';');
            if length(enc_s) == 11
                enc = [tmsp,',', sep(enc_s{3}) ,',', sep(enc_s{7}),',', sep(enc_s{4}),',',
                sep(enc_s{8}),',', sep(enc_s{5}),',', sep(enc_s{9}),',', sep(enc_s{6}),',',
                sep(enc_s{10})];
                fprintf(fenc, '%s \n', enc);

                t = [str2double(tmsp) str2double(sep(enc_s{3})) str2double(sep(enc_s{7}))
                str2double(sep(enc_s{4})) str2double(sep(enc_s{8}))
                str2double(sep(enc_s{5})) str2double(sep(enc_s{9}))
                str2double(sep(enc_s{6})) str2double(sep(enc_s{10}))];
                fencm = [fencm;t];
                %fencm(l,:) = t;
            end
            % lectura accel !
            acel_s = strsplit(spl{3},';');
            if length(acel_s) == 9
                acl = [tmsp,',', sep(acel_s{3}) ,',', sep(acel_s{4}),',',
                sep(acel_s{5}),',', sep(acel_s{6}),',', sep(acel_s{7}),',', sep(acel_s{8})];
                fprintf(facl, '%s \n', acl);

                t = [str2double(tmsp) str2double(sep(acel_s{3})) str2double(sep(acel_s{4}))
                str2double(sep(acel_s{5})) str2double(sep(acel_s{6}))
                str2double(sep(acel_s{7})) str2double(sep(acel_s{8}))];
                faclm = [faclm;t];
                %faclm(l,:) = t;
            end
            % lectura analog !
            ana_s = strsplit(spl{4},';');
            if length(ana_s) == 20
                ana = [tmsp,',', sep(ana_s{3}) ,',', sep(ana_s{4}),',', sep(ana_s{5}),',',
                sep(ana_s{6}),',', sep(ana_s{7}),',', sep(ana_s{8}),',', sep(ana_s{9}),',',
                sep(ana_s{10}),',', sep(ana_s{11}),',', sep(ana_s{12}),',',
                sep(ana_s{13}),',', sep(ana_s{14}),',', sep(ana_s{15}),',',
                sep(ana_s{16}),',', sep(ana_s{17}),',', sep(ana_s{18}),',', sep(ana_s{19})];
                fprintf(fana, '%s \n', ana);

                t = [str2double(tmsp) str2double(sep(ana_s{3})) str2double(sep(ana_s{4}))
                str2double(sep(ana_s{5})) str2double(sep(ana_s{6}))
```

```
        str2double(sep(ana_s{7})) str2double(sep(ana_s{8}))
        str2double(sep(ana_s{9})) str2double(sep(ana_s{10}))
        str2double(sep(ana_s{11})) str2double(sep(ana_s{12}))
        str2double(sep(ana_s{13})) str2double(sep(ana_s{14}))
        str2double(sep(ana_s{15})) str2double(sep(ana_s{16}))
        str2double(sep(ana_s{17})) str2double(sep(ana_s{18}))
        str2double(sep(ana_s{19}))];
        fanam = [fanam;t];
        %fanam(l,:) = t;
    end
    % lectura dig
    dig_s = strsplit(spl{5},';');
    if length(dig_s) == 11
        dig = [tmsp,',', sep(dig_s{3}) ,',', sep(dig_s{4}),',', sep(dig_s{5}),',',
        sep(dig_s{6}),',', sep(dig_s{7}),',', sep(dig_s{8}),',', sep(dig_s{9}),',',
        sep(dig_s{10}),',', sep(dig_s{11})];
        fprintf(fdig, '%s \n', dig);

        t = [str2double(tmsp) str2double(sep(dig_s{3})) str2double(sep(dig_s{4}))
        str2double(sep(dig_s{5})) str2double(sep(dig_s{6}))
        str2double(sep(dig_s{7})) str2double(sep(dig_s{8}))
        str2double(sep(dig_s{9})) str2double(sep(dig_s{10}))
        str2double(sep(dig_s{11}))];
        fdigm = [fdigm;t];
        %fdigm(l,:) = t;
    end
    % lectura lds
    lds_s = strsplit(spl{6},';');

    %display(lds_s);
    %display(length(lds_s));

    if length(lds_s) == 363
        lds_d = [tmsp,','];
        lds_i = [tmsp,','];
        lds_e = [tmsp,','];

        te = [str2double(tmsp)];
        tq = [str2double(tmsp)];
        td = [str2double(tmsp)];

        for i = 3:362
            tmp = strsplit(lds_s{i},',');

            %display(tmp);

            lds_d = [lds_d,tmp{2} ,','];
            lds_i = [lds_i,tmp{3} ,','];
            lds_e = [lds_e,tmp{4} ,','];

            te = [te str2double(tmp{4})];
            tq = [tq str2double(tmp{3})];
            td = [td str2double(tmp{2})];
        end

        flds_errm = [flds_errm;te];
        flds_qualm = [flds_qualm;tq];
```

```
                flds_dism = [flds_dism;td];

                %flds_errm(1,:) = te;
                %flds_qualm(1,:) = tq;
                %flds_dism(1,:) = td;

                fprintf(flds_err, '%s \n', lds_e);
                fprintf(flds_qual, '%s \n', lds_i);
                fprintf(flds_dis, '%s \n', lds_d);
            end

        elseif length(spl) == 3
            % lectura encoders
            enc_s = strsplit(spl{2},';');
            enc = [tmsp,',', sep(enc_s{3}) ,',', sep(enc_s{7}),',', sep(enc_s{4}),',',
            sep(enc_s{8}),',', sep(enc_s{5}),',', sep(enc_s{9}),',', sep(enc_s{6}),',',
            sep(enc_s{10})];
            fprintf(fenc, '%s \n', enc);

            t = [str2double(tmsp) str2double(sep(enc_s{3})) str2double(sep(enc_s{7}))
            str2double(sep(enc_s{4})) str2double(sep(enc_s{8})) str2double(sep(enc_s{5}))
            str2double(sep(enc_s{9})) str2double(sep(enc_s{6})) str2double(sep(enc_s{10}))];
            fencm = [fencm;t];
            %fencm(l,:) = t;
        else
        end
    end
    tline = fgets(ofl);
    l=l+1;
    waitbar(l/nlines,wt,'Generating files...');
end

% Se ponen los 0's del laser a NaN
flds_dism(flds_dism==0)=NaN;

fclose(ofl);

fclose(fenc);
fclose(facl);
fclose(fana);
fclose(fdig);
fclose(flds_err);
fclose(flds_qual);
fclose(flds_dis);

close(wt);

% Se crean los .mat
mfenc = strcat(path,filesep,'data_enc.mat'); % destino encoders
mfacl = strcat(path,filesep,'data_acl.mat'); % destino acel
mfana = strcat(path,filesep,'data_ana.mat'); % destino analog
mfdig = strcat(path,filesep,'data_dig.mat'); % destino digital
mflds_err = strcat(path,filesep,'lds_err.mat'); % destino err_lidar
mflds_qual = strcat(path,filesep,'lds_qua.mat'); % destino qua_lidar
mflds_dis = strcat(path,filesep,'lds_dis.mat'); % destino dis_lidar

save(mfenc,'fencm'); % Se guarda la variable en el archivo
```

```
save(mfacl,'faclm'); % Se guarda la variable en el archivo
save(mfana,'fanam'); % Se guarda la variable en el archivo
save(mfdig,'fdigm'); % Se guarda la variable en el archivo
save(mflds_err,'flds_errm'); % Se guarda la variable en el archivo
save(mflds_qual,'flds_qualm'); % Se guarda la variable en el archivo
save(mflds_dis,'flds_dism'); % Se guarda la variable en el archivo

%----------------- FUNCIONES --------------------
    function s = sep(str)
        pos = strfind(str,',');
        s = str(pos+1:length(str));
    end

end
```

```
function calc_pw(ldx,ldy,trajec,path)

wt = waitbar(0,'Calculating coords...');
pw = zeros(length(ldx),2,360);
[siz,l] = size(ldx);
for i=1:siz
    if ~ishandle(wt)
        break;
    end

    ang = trajec(i,3);

    % Transformacion: laser -> robot -> mundo
    rot_lsr = [ cos(pi/2)  -sin(pi/2)  0;...
        sin(pi/2)   cos(pi/2)  0;...
        0           0          1];

    tra_lsr = [ 1    0    0;...
                0    1    -0.095;...
                0    0    1];

    rot_rbt = [ cos(ang)  -sin(ang)  0;...
                sin(ang)  cos(ang)  0;...
                0         0         1];

    tra_rbt = [ 1    0    trajec(i,1);...
                0    1    trajec(i,2);...
                0    0    1];

    p = [ ldx(i,:);...
          ldy(i,:);...
          ones(1,360) ];

    pwi =  tra_rbt * rot_rbt * tra_lsr * rot_lsr * p;
    pw(i,1:2,:) = pwi (1:2,:);

    waitbar(i/length(ldx),wt,'Calculating coords...');
end
if ishandle(wt)
    close(wt);
end

% Cambia las dimensiones en (x,y ; 360 lecturas ; n lecturas en tiempo)
pw = permute(pw,[2 3 1]);

assignin('base', 'pw', pw);
pwf = strcat(path,filesep,'pw.mat');
save(pwf,'pw');

end
```

Anexo E

Código en el modelo Simulink

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\simulink\neato_interface.cpp miércoles, 21 de octubre de 2015 22:30

```
#define WIN32_LEAN_AND_MEAN

#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <iostream>
#include <chrono>
#include <vector>
#include <exception>

// Need to link with Ws2_32.lib, Mswsock.lib, and Advapi32.lib
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")

using namespace std;

#define DEFAULT_R_BUFLEN 9216
#define DEFAULT_S_BUFLEN 2048
#define PI 3.14159

// ================================================================
// ================================================================
// ============== Class definition code above =====================
// ================================================================
// ================================================================

class socket_client{
    private:
        int sock;
        struct sockaddr_in server;
        SOCKET ConnectSocket;
        int iResult;
        char recvbuf[DEFAULT_R_BUFLEN];
        int find_chr(char, char*, int);
    public:
        socket_client();
        bool conn(string, string);
        string receive();
        bool send_data(string);
        bool close();
        vector<string> split_str(string, char);
        vector<double> ls_d;
        vector<double> ls_q;
};

socket_client::socket_client(){
    sock = -1;
    ConnectSocket = INVALID_SOCKET;
    iResult = 0;
}

//  Splits a string in a vector of strings, using the input character
vector<string> socket_client::split_str(string istr, char chr){
```

```
    vector<string> sol;
    size_t pi = 0;
    size_t pf = istr.find(chr);
    while(pf != -1){
        sol.push_back(istr.substr(pi,pf-pi));
        pi = pf+1;
        pf = istr.find(chr,pi);
    }
    sol.push_back(istr.substr(pi,sol.size()-pi-1)); // Adding the last portion behind the
    last ';'
    return sol;
}

//   Looks for a char in a char array and returns the position (-1 if not found)
int socket_client::find_chr(char cfnd, char* chv, int sz){
    int sal = -1;
    //cout << "char- " << (int)cfnd << " size: " << sz  << '\n';
    for (int l = 0; (l < sz) && (sal == -1); l++){
        //cout << (int)chv[l] << ' ';
        if(chv[l] == cfnd){sal = l;}
    }
    return sal;
}

//    Connect to a host on a certain port number
bool socket_client::conn(string address = "127.0.0.1" , string port = "20000"){
    WSADATA wsaData;
    struct addrinfo *result = NULL, *ptr = NULL, hints;
    // Initialize Winsock
    iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
    if (iResult != 0) { // If 0 OK
        printf("WSAStartup failed with error: %d\n", iResult);
        return false;
    }

    ZeroMemory( &hints, sizeof(hints) );
    hints.ai_family = AF_INET; //AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;

    // Resolve the server address and port (Check IP+por)
    iResult = getaddrinfo(address.c_str(), port.c_str(), &hints, &result);
    if ( iResult != 0 ) {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return false;
    }

    // Attempt to connect to an address until one succeeds
    for(ptr=result; ptr != NULL ;ptr=ptr->ai_next) {
        // Create a SOCKET for connecting to server
        ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype, ptr->ai_protocol);
        if (ConnectSocket == INVALID_SOCKET) {
            printf("socket failed with error: %ld\n", WSAGetLastError());
            WSACleanup();
            return false;
        }
```

```
        // Connect to server.
        iResult = connect( ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
        if (iResult == SOCKET_ERROR) {
            closesocket(ConnectSocket);
            ConnectSocket = INVALID_SOCKET;
            continue;
        }
        break;
    }

    freeaddrinfo(result);

    if (ConnectSocket == INVALID_SOCKET) {
        printf("Unable to connect to server!\n");
        WSACleanup();
        return false;
    }
    return true;
}


//    Receive data from the connected host
string socket_client::receive(){
    // Receive until read new line
        bool end = false;
        string recvtxt = "";

        while(!end){
            memset(recvbuf, 0, sizeof(recvbuf)); // Buffer reset
            iResult = recv(ConnectSocket, recvbuf, DEFAULT_R_BUFLEN, 0);
            if ( iResult > 0 ){
                if( int p = find_chr(char(10), recvbuf, iResult) == -1){
                    recvtxt = recvtxt + string(recvbuf);
                }
                else{
                    recvtxt = recvtxt + string(recvbuf);
                    end = true;
                    return recvtxt;
                }

            }
            else if ( iResult == 0 ){
                printf("Connection closed\n");
                end = true;
            }
            else{
                printf("recv failed with error: %d\n", WSAGetLastError());
                end = true;
            }
        }

    return recvtxt;
}

//    Send data to the connected host
bool socket_client::send_data(string data){
```

```
    data = data + '\n';
    size_t dlen = data.size();
    char sendbuf[DEFAULT_S_BUFLEN];
    copy(data.begin(),data.end(),sendbuf);
     // Send an initial buffer
    iResult = send( ConnectSocket, sendbuf, dlen, 0 );
    if (iResult == SOCKET_ERROR) {
        printf("send failed with error: %d\n", WSAGetLastError());
        closesocket(ConnectSocket);
        WSACleanup();
        return false;
    }
    return true;
}

bool socket_client::close(){

     // shutdown the connection since no more data will be sent
    iResult = shutdown(ConnectSocket, SD_SEND);
    if (iResult == SOCKET_ERROR) {
        printf("shutdown failed with error: %d\n", WSAGetLastError());
        closesocket(ConnectSocket);
        WSACleanup();
        return 1;
    }

     // cleanup
    closesocket(ConnectSocket);
    WSACleanup();
}

// ============================================================================
// ============================================================================
// ============ Simulink specific code above ==================================
// ============================================================================
// ============================================================================

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME  neato_interface

/*
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */
#include "simstruc.h"

#define IS_PARAM_DOUBLE(pVal) (mxIsNumeric(pVal) && !mxIsLogical(pVal) &&\
!mxIsEmpty(pVal) && !mxIsSparse(pVal) && !mxIsComplex(pVal) && mxIsDouble(pVal))


/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S){
    // 2 expected parameters
```

```
    ssSetNumSFcnParams(S, 2);

    //====================================================================
    //====================================================================

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 1);

    // ==== Specify I/O ====
    // Input ports
    if (!ssSetNumInputPorts(S, 2)) return;

    ssSetInputPortDirectFeedThrough(S,0,1);
    ssSetInputPortDirectFeedThrough(S,1,1);

    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortDataType(S, 0, SS_DOUBLE);
    ssSetInputPortWidth(S, 1, 1);
    ssSetInputPortDataType(S, 1, SS_DOUBLE);

    // Output ports
    if (!ssSetNumOutputPorts(S, 4)) return;
    ssSetOutputPortWidth(S, 0, 360);
    ssSetOutputPortDataType(S, 0, SS_DOUBLE);
    ssSetOutputPortWidth(S, 1, 360);
    ssSetOutputPortDataType(S, 1, SS_DOUBLE);
    ssSetOutputPortWidth(S, 2, 4);
    ssSetOutputPortDataType(S, 2, SS_DOUBLE);
    ssSetOutputPortWidth(S, 3, 4);
    ssSetOutputPortDataType(S, 3, SS_DOUBLE);

    ssSetNumSampleTimes(S, 1); // Defines how many sample times are defined in
    (mdlInitializeSampleTimes)
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 1); // reserve element in the pointers vector
    ssSetNumModes(S, 0); // to store a C++ object
    ssSetNumNonsampledZCs(S, 0);

    ssSetSimStateCompliance(S, USE_CUSTOM_SIM_STATE);

    ssSetOptions(S, 0);
}



/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    This function is used to specify the sample time(s) for your
 *    S-function. You must register the same number of sample times as
 *    specified in ssSetNumSampleTimes.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
```

```
#if defined(ssSetModelReferenceSampleTimeDefaultInheritance)
  ssSetModelReferenceSampleTimeDefaultInheritance(S);
#endif

}

#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
  /* Function: mdlStart =======================================================
   * Abstract:
   *    This function is called once at start of model execution. If you
   *    have states that should be initialized once, this is the place
   *    to do it.
   */
  static void mdlStart(SimStruct *S){
        string txt;
        bool conn = false;

        ssGetPWork(S)[0] = (void *) new socket_client;  // store new C++ object in the
        pointers vector
        socket_client *sockt = (socket_client *) ssGetPWork(S)[0];

        char_T buffer[20];
        mxGetString(ssGetSFcnParam(S, 0), buffer, 20);
        string ip = buffer;
        mxGetString(ssGetSFcnParam(S, 1), buffer, 20);
        string port = buffer;

        sockt->conn(ip , port);

        // ======= Communication initialization =======
        // Read Ok serial port
        txt = string(sockt->receive());
        size_t found = txt.find("OK");
        if (found!=std::string::npos){
            cout << "Serial port (Raspberry-Neato) OK" << '\n';
            conn = true;
        }
        else{
            cout << "Serial port (Raspberry-Neato) not found, please check cable connection
            and Neato power on, and restart Raspberry" << '\n';
        }
        // ======= END - Communication initialization =======

        if(conn){
            // ======= Set Datamode Listener =======
            sockt->send_data("DataMode listener");
            txt = string(sockt->receive());

            // Wait 3 seconds
            _sleep(3000);

        }
        // ======= END - Set Datamode Listener =======

  }
#endif /*  MDL_START */
```

```
/* Function: mdlOutputs =======================================================
 * Abstract:
 *    In this function, you compute the outputs of your S-function
 *    block.
 */
static void mdlOutputs(SimStruct *S, int_T tid){
    // Pointer to the instance of the class socket_client
    socket_client *sockt = (socket_client *) ssGetPWork(S)[0];

    // Inputs
    InputRealPtrsType speed_p = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType theta_p = ssGetInputPortRealSignalPtrs(S,1);

    double speed = *speed_p[0];
    double theta = *theta_p[0];

    // Outputs
    real_T *laser_d = ssGetOutputPortRealSignal(S,0);
    real_T *laser_q = ssGetOutputPortRealSignal(S,1);
    real_T *lwheel = ssGetOutputPortRealSignal(S,2);
    real_T *rwheel = ssGetOutputPortRealSignal(S,3);


    // Commands $ robot interactionm
    // ======= Send commands ===========
    // --- Movement ----
    // Calculo de distancia por rueda en funcion de theta y speed (Ajuste de speed??)
    int w = 243; // Wheel axes distance [mm]
    int inc_speed = 30; // A tenth of the max linear speed [mm/s]
    double time_for_a_turn = (2 * PI * w) / (inc_speed * 10); // Robot perimeter divided by
    max_speed (300) [sec] (5.0894)


    // theta correction
    if(theta > 0){theta = fmod(theta,(2 * PI));}
    else{theta = fmod(theta,-(2 * PI));}
    ssPrintf("theta: %f \n",theta);

    double arc_c = speed * time_for_a_turn; // Robot center arc displacement of robot center
    double alpha = theta * time_for_a_turn;  // Incremental angular rotation while moving
    [rad]

    int r = int(arc_c + (w * alpha));
    int l = int(arc_c - (w * alpha));

    ssPrintf("speed: %f \n",speed);

    // ---- Parameters correction --------
    // Speed
    if(speed < 0){speed = speed * -1;}
    if(speed > 300){speed = 300;}

    // R & L distances
    if (abs(r) > 3000){r = r/3;}
    if (abs(l) > 3000){l = l/3;}
```

```
    //-------------------------------------

    char cmmd_buf[100];
    string txt = "";
    sprintf(cmmd_buf,"SetMotor LWheelDist %d RWheelDist %d Speed %f",l,r,speed);
    string cmmd(cmmd_buf);

    ssPrintf("\n%s\n",cmmd_buf);
    sockt->send_data(cmmd);
    txt = sockt->receive();

    // --- Get Laser data ----
    cmmd = "GetLDSScan";
    sockt->send_data(cmmd);
    txt = sockt->receive();
    //cout << txt << '\n';

    ssPrintf(txt.c_str());

    // -- Parsing laser data ---
    vector<string> ret = sockt->split_str(txt, ';');
    vector<string> tmp;
    for(int k=1; k < 361;k++){ //Angle, Dist , Quality, Error
        tmp = sockt->split_str(ret[k], ',');
        laser_d[k-1] = atof(tmp[1].c_str());
        laser_q[k-1] = atof(tmp[2].c_str());
        tmp.clear();
    }

    // --- Get Wheels data ---
    // Send command until get vaild response (sometimes no data (??))
    bool repeat = true;
    cmmd = "GetMotors LeftWheel RightWheel";
    while (repeat){
        sockt->send_data(cmmd);
        txt = sockt->receive();
        if(txt.length() > 200){repeat = false;}
    }

    ssPrintf(txt.c_str());

    // --- Parsing wheels data ---
    ret.clear();
    ret = sockt->split_str(txt, ';');

    for(int k=2; k < 10;k++){ //Angle, Dist , Quality, Error
        tmp = sockt->split_str(ret[k], ',');
        if(k < 6){
            lwheel[k-2] = atof(tmp[1].c_str());
            }
        else{
            rwheel[k-6] = atof(tmp[1].c_str());
        }
        tmp.clear();
    }
}
```

```
/* Function: mdlTerminate =====================================================
 * Abstract:
 *    In this function, you should perform any actions that are necessary
 *    at the termination of a simulation.  For example, if memory was
 *    allocated in mdlStart, this is the place to free it.
 */
static void mdlTerminate(SimStruct *S){
    // Pointer to the instance of the class socket_client
    socket_client *sockt = (socket_client *) ssGetPWork(S)[0];

    sockt->send_data("Close");
    sockt->close();
    delete sockt;

}
/*=============================================================*
 * See sfuntmpl.doc for the optional S-function methods *
 *=============================================================*/

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

```
%sldebug sl_drivepoint

function [x,y,ntheta] = fcn(ticsl,ticsr,x0)

persistent x_m;
persistent y_m;
persistent theta_m;
persistent old_ticsl;
persistent old_ticsr;

% Persistent vars initialization
if isempty(x_m)
   x_m = x0(1);
   y_m = x0(2);
   theta_m = x0(3);
   old_ticsl = ticsl;
   old_ticsr = ticsr;
end

w = 243; % Wheel axes distance [mm]
i_tics_l = ticsl - old_ticsl; % Lwheel tics increment
i_tics_r = ticsr - old_ticsr; % Rwheel tics increment

% === Data process ===
% Theta increment
if (i_tics_r == i_tics_l)

else
    % Keep new theta value in memory
    itheta = (i_tics_r - i_tics_l) / w;
    theta_m = theta_m + itheta;
end

% New position
despl = (i_tics_l + i_tics_r) / 2;
ix = (cos(theta_m) * despl);
iy = (sin(theta_m) * despl);

% Keep current position in memory
x_m = x_m + ix;
y_m = y_m + iy;

% Update old tics values for the next iteration
old_ticsl = ticsl;
old_ticsr = ticsr;

% Output current position and theta
x = x_m / 1000; % [mm] to [m]
y = y_m / 1000; % [mm] to [m]
ntheta = theta_m; % [rads]
```

Anexo F

Código interfaz web

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\web\NeatoCmmd.html miércoles, 21 de octubre de 2015 22:43

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <style>
            #main{
                width: 1250px;
            }
            #left {
                float:left;
                width:500px;
            }
            #right {
                float:left;
                width:750px;
            }
            #l_bott {
                border: 1px solid black;
                position: absolute;
                top: 530px;
                left: 8px;

                width: 499px;
                /*clear:both;*/
            }
            #bottom {
                /*border: 1px solid black;*/
                clear:both;
            }
            #url {
                resize:none;
            }
            #inputtext {
                margin-top: 5px;
                resize:none;
            }
            #outputtext {
                resize:none;
            }
            #g_map{
                /*border: 1px solid black;*/
            }
        </style>
    </head>

    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
    <script src="js/Radar.js"></script>
    <script src="js/GridMap.js"></script>
    <script src="js/RData.js"></script>
    <script src="js/RControl.js"></script>
    <script src="js/main.js"></script>

    <body>
        <div id="main">

            <div id="wrapper">
                    <div id="left">
```

```
                        <!-- ############### RADAR ############### -->
                        <canvas id="g_radar" width="500" height="500">
                    </div>

                    <div id="right">
                        <!-- ############### GRID MAP ############### -->
                        <canvas id="g_map" width="750" height="750">
                        This text is displayed if your browser does not support HTML5
                        Canvas.</canvas>

                        <button id="z_out">Zoom -</button>
                        <button id="z_in">Zoom +</button>
                        <input type="checkbox" id="chk_all">Show accumulated previous world
                        points<br>
                    </div>

                    <div id="l_bott">
                        <textarea id="url" rows="1" cols="59"></textarea>
                        <button id="connectButton">Connect</button>
                        <button id="disconnectButton">Disconnect</button>
                        <textarea id="inputtext" cols="59"></textarea>
                        <button id="sendButton">Send</button>
                        <button id="clearButton">Clear</button>


                    </div>

            </div>

            <div id="bottom">
                <textarea id="outputtext" rows="5" cols="153"></textarea>
            </div>

        </div>
    </body>
</html>
```

```
// Executes on page load
$(function () {
    // ##### File index control #####
    // ################################
    $dat_in = 0;

    // ######### Websocket ##########
    // ################################
    $wskt = null;

    // ############ Data ############
    // ################################
    $dat = new RData();

    // ####### Robot Control ########
    // ################################
    $ctrl = new RControl();

    // ######## Control Panel #######
    // ################################
    init_controls();

    // ######### Grid Map ###########
    // ################################
    $map = new GridMap('g_map','#z_in','#z_out','#z_center','#chk_all');


    // ######### Radar Plot ######
    // ################################
    // Create radar plot
    $radar = new Radar('g_radar');


})


    function init_controls()
    {
        $('#inputtext').val("String to be sent to the robot.");
        $('#url').val("ws://192.168.1.12:20000/");
        $('#disconnectButton').prop( "disabled", true );
        $('#sendButton').prop( "disabled", true );
        $('#clearButton').prop( "disabled", true );
        $('#inputtext').prop( "disabled", true );
        $('#outputtext').prop( "disabled", true );

        // ####### Control Buttons ######
        // ################################
        $('#connectButton').click(function() {
            $wskt = new WebSocket($('#url').val());
            $wskt.onopen = function(evt) { onOpen(evt) };
            $wskt.onclose = function(evt) { onClose(evt) };
            $wskt.onmessage = function(evt) { onMessage(evt) };
            $wskt.onerror = function(evt) { onError(evt) };

            $ctrl.skt = $wskt;
```

```
        });

        $('#disconnectButton').click(function() { //Enviar Close
            //$wskt.close();
            $ctrl.stopWRKR();
            $wskt.send("Close");
            $wskt.close();
            onClose();
        });

        $('#sendButton').click(function() {
            message = $('#inputtext').val();
            writeToScreen("sent: " + message + '\n');
            $wskt.send(message);
        });

        $('#clearButton').click(function() {
            $('#outputtext').val('');
        });

    }

    function onOpen(evt)
    {
        writeToScreen("connected\n");
        $('#disconnectButton').prop( "disabled", false );
        $('#sendButton').prop( "disabled", false );
        $('#clearButton').prop( "disabled", false );
        $('#inputtext').prop( "disabled", false );
        $('#outputtext').prop( "disabled", false );
        $('#connectButton').prop( "disabled", true );
        $('#url').prop( "disabled", true );
    }
    function onClose(evt)
    {
        writeToScreen("disconnected\n");
        $('#disconnectButton').prop( "disabled", true );
        $('#sendButton').prop( "disabled", true );
        $('#clearButton').prop( "disabled", true );
        $('#inputtext').prop( "disabled", true );
        $('#outputtext').prop( "disabled", true );
        $('#connectButton').prop( "disabled", false );
        $('#url').prop( "disabled", false );

        showJSON();
    }
    function onMessage(evt)
    {
        writeToScreen("response: " + evt.data + '\n');
        if($dat.addData(evt.data)){ // If new data processed
            // Update radar & grid
            $radar.updtGrph($dat.p_laser[$dat.vn-1][0],$dat.p_laser[$dat.vn-1][1]);
            $map.updtGrph($dat); // Map Update

        }

    }
```

```
    function onError(evt)
    {
        writeToScreen('error: ' + evt.data + '\n');
        $wskt.close();
        $('#connectButton').prop( "disabled", false );
        $('#disconnectButton').prop( "disabled", true );
    }


    function writeToScreen(message)
    {
        $('#outputtext').val( $('#outputtext').val() + message  )
        $('#outputtext').scrollTop($('#outputtext').attr('scrollHeight'));
        //.get(0).scrollHeight
    }


    function showJSON(){
        var recipe =  window.open('','RecipeWindow','width=600,height=600');
        var html = '<html><body><pre>' + JSON.stringify($dat); + '</pre></body></html>';
        recipe.document.open();
        recipe.document.write(html);
        recipe.document.close();
    }
```

```
var RData = (function () {

    function RData() {
    //init : function () { // Initialize data structure (first element zero values)
        this.vn = 1; // Keeps the last used index updated (values number)
        this.pd = 0; // Keeps las used index with environmental data
        this.timestamp = new Array(); // Keeps timestamp information [Array]
        this.timestamp.push(0);
        this.pose = new Array(); // Keeps robot position and orientation information
        [Array[Array(x,y,theta)]] [mm,mm,]
        this.pose.push(new Float32Array(3));
        this.pose[0][0] = 0;
        this.pose[0][1] = 0;
        this.pose[0][2] = 0;
        this.p_laser = new Array(null); //[[[ok],[nok]],]// Keeps robot environment detected
        points at each position (laser coordinates, angle-distance [2 vectors, ok and error])
        this.p_env = new Array(null); // Keeps robot environment detected points at each
        position (world coordinates, using kd trees)
        this.allp_env = null; // Makes a kd tree with all the world points

        this.r_data = {
            wheels : new Array([0,0,0,0,0,0,0,0]), //
            [l_rpm,l_load,l_pos,l_sped,r_rpm,r_load,r_pos,r_sped] x8
            laser : new Array(null), //Laser  data [distance, quality ,error] [360]
            accel : new Array(null), // [pitch,roll,x,y,z,sum] x6
            analog : new Array(null), //
            [wall_S,batt_v,l_drop,r_drop,l_mag,r_mag,ui_btt,vac_curr,charge,bat_temp1,bat_temp
            2,current, side_b_c,volt_ref,accx,accy,accz] x17
            digital : new Array(null) // [dc_jack,dust_bin,lwe,rwe,lsb,lfb,rsb,rfb] (2x
            wheels extended, 4, bumpers) x8
        }
    }


    // ================================================
    // ======= Clones data from another object ==========
    // ================================================
    RData.prototype.Clone = function(obj) {
        this.vn = obj.vn;
        this.pd = obj.pd;
        this.timestamp = obj.timestamp;
        this.pose = obj.pose;
        this.p_laser = obj.p_laser
        this.p_env = obj.p_env ;
        this.allp_env = obj.allp_env;
        this.r_data = obj.r_data;
    }



    RData.prototype.addData = function(toParse) {
        stat = true;
        error = toParse.indexOf('##') != -1;

        if(toParse.indexOf('GetAccel') != -1 && toParse.length > 5600 && !error){ // Is a
        complete data string
            bdat = toParse.split('#'); // time, motors, accel, analog, digital, laser
```

```
            this.timestamp.push(parseFloat(bdat[0]).toFixed(12));

            mdat = bdat[1].split(';');
            acdat = bdat[2].split(';');
            andat = bdat[3].split(';');
            ddat = bdat[4].split(';');
            ldat = bdat[5].split(';');

            vmdat = Array(); // Motors
            vacdat = Array(); // Accel
            vandat = Array(); // Analog
            vddat = Array(); // Digital
            vldat = Array(); // Laser

            // Get data and fulfill vectors
            for(i=2; i < ldat.length -2 ; i++){
                if(i < mdat.length-1){
                    tt = mdat[i].split(',');
                    vmdat.push(parseInt(tt[1]));
                }
                if(i < acdat.length-1){
                    tt = acdat[i].split(',');
                    vacdat.push(parseInt(tt[1]));
                }
                if(i < andat.length-1){
                    tt = andat[i].split(',');
                    vandat.push(parseInt(tt[1]));
                }
                if(i < ddat.length-1){
                    tt = ddat[i].split(',');
                    vddat.push(parseInt(tt[1]));
                }
                tt = ldat[i].split(',');
                vt = Array(parseInt(tt[1]),parseInt(tt[2]),parseInt(tt[3])); // dist,
                quality, error (error if != 0)

                vldat.push(vt); // Add Laser data
            }
            // Put data vectors in struct
            this.r_data.wheels.push(vmdat);
            this.r_data.laser.push(vldat);
            this.r_data.accel.push(vacdat);
            this.r_data.analog.push(vandat);
            this.r_data.digital.push(vddat);

            // Get current robot position and orientation
            this._getPose();

            // Get world points coordinates
            this._getWcoords();

            //Update position counter
            this.pd = this.vn;
            this.vn = this.vn + 1;
        }
        else if(toParse.indexOf('GetMotors') != -1 && toParse.length > 200 && !error){ // Is
        a (lone) motors data string
```

```
            bdat = toParse.split('#'); // time, motors
            this.timestamp.push(parseFloat(bdat[0]).toFixed(12));
            mdat = bdat[1].split(';');

            vmdat = Array(); // Motors

            for(i=2; i < mdat.length -1 ; i++){
                tt = mdat[i].split(',');
                vmdat.push(parseInt(tt[1]));
            }

            // Put data vectors in struct
            this.r_data.wheels.push(vmdat);
            this.r_data.laser.push(null);
            this.r_data.accel.push(null);
            this.r_data.analog.push(null);
            this.r_data.digital.push(null);

            // Get current robot position and orientation
            this._getPose();

            // Full empty laser data
            this.p_laser.push(null);
            this.p_env.push(null);

            //Update position counter
            this.vn = this.vn + 1;
        }
        else{ // Nothing to treat
            stat = false;
        }
        console.log(this);
        return stat;
    }


    // ================================================
    // ============== Updates robot pose ==============
    // ================================================
    RData.prototype._getPose = function() {
        // Use this.vn -1 as old pose and motors data
        old = this.vn -1;
        w = 243; // Wheel axes distance [mm]
        i_tics_l = this.r_data.wheels[old+1][2] - this.r_data.wheels[old][2]; // Lwheel tics
        increment (2,6)
        i_tics_r = this.r_data.wheels[old+1][6] - this.r_data.wheels[old][6]; // Rwheel tics
        increment
        i_theta = 0;

        // Theta increment
        if(i_tics_l != i_tics_r){
            i_theta = ((i_tics_r - i_tics_l) / w);
        }

        n_theta = this.pose[old][2] + i_theta;

        // New position
```

```
        despl = ((i_tics_l + i_tics_r) / 2);
        iy = (Math.cos(n_theta) * despl) / 1000; //[mm] -> [m]
        ix = -(Math.sin(n_theta) * despl) / 1000; //[mm] -> [m]

        n_x = (this.pose[old][0] + ix); //[m]
        n_y = (this.pose[old][1] + iy); //[m]

        // Save new pose
        this.pose.push(new Array(n_x,n_y,n_theta)); //[mm,mm,rad]
    }


    // ====================================================
    // === Gets world coordinates for a laser point =====
    // === and generates laser coordinates points v =====
    // ====================================================
    RData.prototype._getWcoords = function() {
        t = new Array(); // temporal array for environment points
        lok = new Array(); //[ang,dist]
        lnok = new Array(); //[ang,dist]
        //For each laser reading
        for(u =0; u < this.r_data.laser[this.vn].length ; u++){
            d = this.r_data.laser[this.vn][u][0];
            if(this.r_data.laser[this.vn][u][2] == 0 && d < 6005){ // No error
                // Get laser coordinates
                ang = u * (Math.PI/180); // Deg to radians
                xl = (Math.cos(ang) * d) / 1000; //[mm->m]
                yl = (Math.sin(ang) * d) / 1000; //[mm->m]

                // Get world coordinates
                t.push(this._calculateWc(xl,yl));

                // Generate laser coordinates
                lt = new Array(u,d);
                lok.push(lt);

            }
            else{
                lt = new Array(u,6050);
                lnok.push(lt);
            }

        }
        // Save environment points
        this.p_env.push(t);

        // Save laser points
        lsrp = new Array(lok,lnok);
        this.p_laser.push(lsrp);

    }


    // ========================================================
    // ==== Gets world coordinates from laser coordinates ====
    // ========================================================
    RData.prototype._calculateWc = function(xl,yl) {
```

```
        dlaser = -0.095;
        theta_r = this.pose[this.vn][2]; // [radians]
        cth = Math.cos(theta_r);
        sth = Math.sin(theta_r);
        pxr = this.pose[this.vn][0];
        pyr = this.pose[this.vn][1];

        xw = ((-sth) * xl) + ((-cth) * yl) + (-sth * dlaser) + pxr;
        yw = (( cth) * xl) + ((-sth) * yl) + ( cth * dlaser) + pyr;

        return new Array(xw,yw);
    }

    return RData;
})();
```

```
var Radar = (function () {

    function Radar(canvs) {
        // ================================================
        // ============== Global variables ================
        // ================================================
        this.margin = 0.1; // Indicates space left on each side
        this.rmargin = 0.025; // Indicates radar margin %
        this.axeStyle = [2,5,2];
        this.axeColor = '#C0C0C0';

        this.okcolor = '#58C77D';
        this.nokcolor = '#75200B';
        this.lsrStyle = [0];

        this.canvas = document.getElementById(canvs);
        this.context = this.canvas.getContext("2d");

        this.back_rad = (this.canvas.width/2) * (1-this.margin);
        this.step = (this.back_rad * (1-this.rmargin))/6;

        // Set center in the canvas center
        this.context.translate(this.canvas.width/2, this.canvas.height/2);

        this.drawGrid();
    }

    //
    ================================================================================================
    ========
    // ================================== FUNCTIONS
    ======================================================
    //
    ================================================================================================
    ========

    // ================================================
    // ============== Updates map grid ================
    // ================================================
    Radar.prototype.updtGrph = function (ok,nok) {
        this.drawGrid();
        this.drawLines(ok,this.okcolor);
        this.drawLines(nok,this.nokcolor);
    }

    // ================================================
    // ========== Fixes to standard coords ============
    // ================================================
    Radar.prototype.fixCoords = function (x, y) {
        return [x, -y];
    }

    // ================================================
    // ============ Creates radar grid ================
    // ================================================
    Radar.prototype.drawGrid = function () {
```

```
        // Black background
        this.drawCircle(0,0,this.back_rad,'#000000');

        // Distance circles
        this.drawCline(0,0,this.step);
        this.drawCline(0,0,this.step*2);
        this.drawCline(0,0,this.step*3);
        this.drawCline(0,0,this.step*4);
        this.drawCline(0,0,this.step*5);
        this.drawCline(0,0,this.step*6);

        // Axes
        this.drawAxes();

    }


    // ================================================
    // ================ Draws a circle ================
    // ================================================
    Radar.prototype.drawCircle = function (x, y, rad, fcolor) {
        nc = this.fixCoords(x, y);
        this.context.beginPath();
        this.context.arc(nc[0], nc[1], rad, 0, 2 * Math.PI, true);
        if(fcolor != null){
            this.context.fillStyle = fcolor;
            this.context.fill();
        }
        this.context.lineWidth = 1;
        this.context.strokeStyle = fcolor;
        this.context.stroke();
    }

    // ================================================
    // ================ Draws circle axe ==============
    // ================================================
    Radar.prototype.drawCline = function (x, y, rad) {
        nc = this.fixCoords(x, y);
        this.context.beginPath();
        this.context.setLineDash(this.axeStyle);
        this.context.arc(nc[0], nc[1], rad, 0, 2 * Math.PI, true);
        this.context.lineWidth = 1;
        this.context.strokeStyle = this.axeColor;
        this.context.stroke();
    }


    // ================================================
    // ================== Draws axe ===================
    // ================================================
    Radar.prototype.drawAxes = function () {
        d= this.step*6;
        ang = Math.PI/4;
        labld = d + (40 * 17 * this.rmargin);

        this.context.beginPath();
        this.context.setLineDash(this.axeStyle);
```

```
        this.context.moveTo(-d,0);
        this.context.lineTo(d,0);

        this.context.moveTo(0,-d,0);
        this.context.lineTo(0,d,0);

        this.context.moveTo(d * Math.cos(ang),d * Math.sin(ang));
        this.context.lineTo(-d * Math.cos(ang),-d * Math.sin(ang));

        this.context.moveTo(d * Math.cos(-ang),d * Math.sin(-ang));
        this.context.lineTo(-d * Math.cos(-ang),-d * Math.sin(-ang));

        this.context.strokeStyle = this.axeColor;
        this.context.stroke();

        // Write tags
        this.context.font = '15px Calibri';
        this.context.fillStyle = '#7D7D7D';
        this.context.fillText("0", -4, -labld ); //
        this.context.fillText("45", -labld * Math.cos(ang) - 6, -labld * Math.sin(ang) + 4);
        //
        this.context.fillText("90", -labld - 7 , 7 ); //
        this.context.fillText("135", -labld * Math.cos(ang) -15, labld * Math.sin(ang) + 2 );
        this.context.fillText("180", -10, labld ); //
        this.context.fillText("225", labld * Math.cos(ang), labld * Math.sin(ang) ); //
        this.context.fillText("270", labld -9, 5 ); //
        this.context.fillText("315", labld * Math.cos(-ang) - 6, labld * Math.sin(-ang) + 4
        ); //

        this.context.fillText("1m", -22, -this.step + 13 );
        this.context.fillText("2m", -22, -(this.step * 2) + 13 );
        this.context.fillText("3m", -22, -(this.step * 3) + 13 );
        this.context.fillText("4m", -22, -(this.step * 4) + 13 );
        this.context.fillText("5m", -22, -(this.step * 5) + 13 );
        this.context.fillText("6m", -22, -(this.step * 6) + 13 );
    }


    // ================================================
    // ================ Draws a line ==================
    // ================================================
    Radar.prototype.drawLines = function (p_array, color) { //[[ang,dist]]
        l = p_array.length;
        if (l > 1) {
            this.context.beginPath();
            this.context.setLineDash(this.lsrStyle);
            for (i = 0; i < l; i++) {
                d = (p_array[i][1] / 1000) * this.step;
                ang = ((p_array[i][0] / 180) * Math.PI) + (Math.PI/2);
                x = d * Math.cos(ang);
                y = -(d * Math.sin(ang));

                this.context.moveTo(0, 0);
                this.context.lineTo(x,y);
            }
            this.context.strokeStyle = color;
```

```
            this.context.stroke();
        }
    }

    return Radar;
})();
```

```
var GridMap = (function () {

    function GridMap(canvs, zoomp, zooml, zcenter,chk_all) {
        // ================================================
        // ============== Global variables ================
        // ================================================

        this.Hvals = 20;
        this.zoom = 10; // Indicates number of meters/squares for each side (always pair)
        this.margin = 10; // Indicates space left on each side
        //this.canvas = document.getElementById('g_map');
        this.canvas = document.getElementById(canvs);
        //console.log(this.canvas);
        this.context = this.canvas.getContext("2d");
        //console.log(this.context);
        this.map_w = this.canvas.width - (this.margin * 2);
        this.map_h = this.canvas.height - (this.margin * 2);
        this.step = this.map_w / this.zoom;

        this.p_world = []; // Memory for last world painted
        // Load robot image;
        this.rbtimg = new Image();


        //================================================
        // ============== Robot variables ================
        // ================================================
        this.rbt_rel = 0.318; // Robot heigt relation aspect (cm)

        //================================================

        // Set buttons actions
        var self = this;
        this.rbtimg.onload = function () {
            self.updtGrphT();
        };
        this.rbtimg.src = "js/img/neato_top_f.jpg";

        var self = this;

        $(zooml).click(function () {
            self.zoom += 2;
            self.updtGrph(null);
        });

        $(zoomp).click(function () {
            if (self.zoom > 4) {
                self.zoom -= 2;
                self.updtGrph(null);
            }
        });

        $(zcenter).click(function () {

        });

    }
```

```
//
================================================================================================
========
// ================================== FUNCTIONS
=======================================================
//
================================================================================================
========

// ================================================
// ============== Updates map grid ================
// ================================================
GridMap.prototype.updtGrph = function (dat) {

    this.drawGrid();

    if(dat == null){ //No data input
        this._drawWorldn(dat,0);
        old = dat.vn - 1;
        this.drawRobot(dat.pose[old][0], dat.pose[old][1], dat.pose[old][2]);
    }
    else{ // Draw environment
        // Draw world points
        this._drawWorldn(dat,dat.pd);
        // Draw robot path

        // Draw robot
        old = dat.vn - 1;
        this.drawRobot(dat.pose[old][0], dat.pose[old][1], dat.pose[old][2]);
    }

}


GridMap.prototype.updtGrphI = function (dat,i) {
    this.drawGrid();
    if($(chk_all).is(':checked')){
        this._drawWorldH(dat,i);
    }
    else{
        this._drawWorldn(dat,i);
    }
    //console.log(dat);
    this.drawRobot(dat.pose[i][0], dat.pose[i][1], dat.pose[i][2]);
}


GridMap.prototype.updtGrphT = function () {
    this.drawGrid();
    this.drawRobot(0, 0, 0);
}



// ================================================
// ==== Draws world historical laser points =========
```

```
// ================================================
GridMap.prototype._drawWorldH = function (d,p) {
    if(d != null){
        h = p - this.Hvals;
        if(h<1){h=1;}
        for(; h < p; h++){
            points = d.p_env[h];
            if(points != null){
                this.p_world = points;
                for(i=0; i < points.length; i++){
                    this.drawX(points[i][0],points[i][1],'#CA0A0A');
                }
            }
        }
    }
}


// ================================================
// ============= Draws world laser points ==========
// ================================================
GridMap.prototype._drawWorldn = function (d,p) {
    if(d != null){
        points = d.p_env[p];
        if(points != null){
            this.p_world = points;
            for(i=0; i < points.length; i++){
                this.drawX(points[i][0],points[i][1],'#CA0A0A');
            }
        }
    }
}

// ================================================
// ============== Updates map grid ================
// ================================================
GridMap.prototype.fixCoords = function (x, y) {
    return [x * this.step, -y * this.step];
}

// ================================================
// ============== Creates map grid ================
// ================================================
GridMap.prototype.drawGrid = function () {
    // Reset transform matrix
    this.context.setTransform(1, 0, 0, 1, 0, 0);
    this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
    // Update step
    this.step = this.map_w / this.zoom;

    this.context.strokeStyle = '#C0C0C0'; //'#000000' '#7A7A7A'
    this.context.strokeWidth = 0.1;

    this.context.beginPath();
    for (var x = 0; x <= this.canvas.width; x += this.step) {
        this.context.moveTo(x + this.margin, this.margin);
        this.context.lineTo(x + this.margin, this.map_h + this.margin);
```

```
            this.context.moveTo(this.margin, x + this.margin);
            this.context.lineTo(this.map_w + this.margin, x + this.margin);
        }

        this.context.closePath();

        this.context.stroke();


        // === Draw origin axes ==
        this.context.strokeStyle = '#757575'; //'#000000' '#7A7A7A'
        center = this.canvas.width / 2;

        this.context.beginPath();

        this.context.moveTo(center, this.margin);
        this.context.lineTo(center, this.map_h + this.margin);

        this.context.moveTo(this.margin, center);
        this.context.lineTo(this.map_w + this.margin, center);

        this.context.closePath();
        this.context.stroke();

        // === Move canvas origin to center of grid! ===
        this.context.translate(this.map_w / 2 + this.margin, this.map_h / 2 + this.margin);
    }

    // ================================================
    // =============== Draws the robot ================
    // ================================================
    GridMap.prototype.drawRobot = function (x, y, theta) {
        nc = this.fixCoords(x, y);

        nszr = (this.step * this.rbt_rel) / this.rbtimg.height; // new height relation in
        pixels
        v_d = (this.rbtimg.height * nszr);
        h_d = (this.rbtimg.width * nszr);
        turn = -theta;
        npx = nc[0] * Math.cos(-turn) - nc[1] * Math.sin(-turn);
        npy = nc[0] * Math.sin(-turn) + nc[1] * Math.cos(-turn);

        this.context.save();
        this.context.rotate(turn);
        this.context.translate(-h_d / 2, -v_d / 2);
        this.context.drawImage(this.rbtimg, npx, npy, h_d, v_d);
        this.context.restore();
    }

    // ================================================
    // ================= Draws a point ================
    // ================================================
    GridMap.prototype.drawPoint = function (x, y, rad, color) {
        nc = this.fixCoords(x, y);
        this.context.beginPath();
        this.context.arc(nc[0], nc[1], (rad / this.zoom) * 5, 0, 2 * Math.PI, true);
```

```
        this.context.fillStyle = color;
        this.context.fill();
        this.context.lineWidth = 1;
        this.context.strokeStyle = color;
        this.context.stroke();
    }

    // ================================================
    // ================== Draws a X ===================
    // ================================================
    GridMap.prototype.drawX = function (x, y, color) {
        nc = this.fixCoords(x, y);
        this.context.beginPath();
        size = (30 / this.zoom);
        this.context.moveTo(nc[0] - size, nc[1] - size);
        this.context.lineTo(nc[0] + size, nc[1] + size);
        this.context.moveTo(nc[0] + size, nc[1] - size);
        this.context.lineTo(nc[0] - size, nc[1] + size);
        this.context.lineWidth = 1;
        this.context.strokeStyle = color;
        this.context.stroke();
    }

    // ================================================
    // ================ Draws a line ==================
    // ================================================
    GridMap.prototype.drawLine = function (p_array, color) {
        l = p_array.length;
        if (l > 1) {
            nc = this.fixCoords(p_array[0][0], p_array[0][1]);
            this.context.beginPath();
            this.context.moveTo(nc[0], nc[1]);
            for (i = 1; i < l; i++) {
                nc = this.fixCoords(p_array[i][0], p_array[i][1]);
                this.context.lineTo(nc[0], nc[1]);
            }
            this.context.strokeStyle = color;
            this.context.stroke();
        }
    }

    // ================================================
    // ========= Draws segment separator ==============
    // ================================================
    GridMap.prototype.drawSegSep = function (array, color) {

        l = p_array.length;
        if (l > 1) {
            nc = this.fixCoords(p_array[0][0], p_array[0][1]);
            this.context.beginPath();
            this.context.moveTo(nc[0], nc[1]);
            for (i = 1; i < l; i++) {
                nc = this.fixCoords(p_array[i][0], p_array[i][1]);
                this.context.lineTo(nc[0], nc[1]);
            }
            this.context.strokeStyle = color;
            this.context.stroke();
```

```
        }
    }


    // ================================================
    // ============== Draws an ellipse ===============
    // ================================================
    GridMap.prototype.drawEllipse = function (centerX, centerY, width, height) {
        this.context.beginPath();

        this.context.moveTo(centerX, centerY - height/2); // A1

        this.context.bezierCurveTo(
            centerX + width/2, centerY - height/2, // C1
            centerX + width/2, centerY + height/2, // C2
            centerX, centerY + height/2); // A2

        this.context.bezierCurveTo(
            centerX - width/2, centerY + height/2, // C3
            centerX - width/2, centerY - height/2, // C4
            centerX, centerY - height/2); // A1

        this.context.closePath();
    }


    return GridMap;
})();
```

```
var RControl = (function () {

    function RControl() {
        this.skt = null; // Socket to use
        this.w = 243; // Wheel axes distance [mm]
        this.alpha = 0; // Incremental angular rotation while moving [rad]
        this.spd_factor = 0.5; // Speed security factor
        this.speed = 0; // Current speed [mm/s] (max 300)
        this.spd_inc = 30; // A tenth of the max linear speed [mm/s] (up/down arrows)
        this.theta_dot = 0 // Current robot angular speed [rad/seg]
        this.max_theta_dot = 2.51; // [rad/s] 2 pi / time needed for a turn (perimeter / max
        linear speed)
        this.inc_ang_turn = this.max_theta_dot/10; // A tenth of the max angular speed
        [rad/s] (left/right arrows)
        this.time_for_a_turn = (2 * Math.PI * this.w) / (this.spd_inc * 10); // Robot
        perimeter divided by max_speed (300) [sec] (5.0894)

        this.wrkr = null;
        this.msg = '';

        this.setControls();
    }

    // ================================================
    // ==== Defines a worker for command repetition =====
    // ================================================
    RControl.prototype.createWorker = function(t){
        if(this.wrkr != null){
            this.wrkr.terminate();
        }
        this.wrkr = new Worker("js/worker.js");
        self = this;
        this.wrkr.onmessage = function(e){
            console.log('WRK on message');
            console.log(t);
            self.skt.send(self.msg);
        }
        this.wrkr.postMessage(t);
    }


    // ================================================
    // ======= Defines actions taken on keypress ========
    // ================================================
    RControl.prototype.setControls = function() {
        // ######## Keys Capture ########
        // ##############################
        self = this;
        $(document).keydown(function(e) {
            switch (e.keyCode) {
                case 37: // Left
                    self.b_left();
                    console.log(this.msg);
                    break;
                case 38: // Up
                    self.b_up();
                    console.log(this.msg);
```

```
                    break;
                case 39: // Right
                    self.b_right();
                    console.log(this.msg);
                    break;
                case 40: //Down
                    self.b_down();
                    console.log(this.msg);
                    break;
                case 32: // Space
                    self.b_space();
                    console.log(this.msg);
                    break;
            }
        });
    }

    // ================================================
    // ======= Sends left movement command =======
    // ================================================
    RControl.prototype.b_left = function() {
        this.theta_dot = this.theta_dot + (this.inc_ang_turn * this.spd_factor);
        if(this.theta_dot > this.max_theta_dot){
            this.theta_dot = this.max_theta_dot;
        }
        ds = this.turn();
        wr_spd = 0;
        if(this.speed == 0){
            this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
            Math.round(ds[1]).toString() + ' Speed '+
            Math.round(Math.abs(ds[1]/this.time_for_a_turn)).toString();
            wr_spd = Math.round(Math.abs(ds[1]/this.time_for_a_turn));
        }
        else{
            this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
            Math.round(ds[1]).toString() + ' Speed '+
            Math.round(Math.abs(this.speed)).toString();
            wr_spd = Math.round(Math.abs(this.speed));
        }
        d = Math.min(Math.abs(Math.round(ds[0])),Math.abs(Math.round(ds[1])));
        wr_time = Math.round((d/wr_spd)*850).toString();
        this.skt.send(this.msg);
        this.createWorker(wr_time);
        console.log('cmmd - '+this.msg);
    }
    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
    RControl.prototype.b_up = function() {
        if(this.speed < 300){
            this.speed = this.speed + (this.spd_inc * this.spd_factor);
        }
        else{
            this.speed = 300;
        }
        ds = this.turn();
        this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
```

```
        Math.round(ds[1]).toString() + ' Speed '+ Math.round(Math.abs(this.speed)).toString();
        d = Math.min(Math.abs(Math.round(ds[0])),Math.abs(Math.round(ds[1])));
        wr_time = Math.round((d/Math.round(Math.abs(this.speed)))*850).toString();
        this.skt.send(this.msg);
        this.createWorker(wr_time);
        console.log('cmmd - '+this.msg);
    }
    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
    RControl.prototype.b_right = function() {
        this.theta_dot = this.theta_dot - (this.inc_ang_turn * this.spd_factor);
        if(this.theta_dot < -this.max_theta_dot){
            this.theta_dot = -this.max_theta_dot;
        }
        ds = this.turn();
        wr_spd = 0;
        if(this.speed == 0){
            this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
            Math.round(ds[1]).toString() + ' Speed '+
            Math.round(Math.abs(ds[1]/this.time_for_a_turn)).toString();
            wr_spd = Math.round(Math.abs(ds[1]/this.time_for_a_turn));
        }
        else{
            this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
            Math.round(ds[1]).toString() + ' Speed '+
            Math.round(Math.abs(this.speed)).toString();
            wr_spd = Math.round(Math.abs(this.speed));
        }
        d = Math.min(Math.abs(Math.round(ds[0])),Math.abs(Math.round(ds[1])));
        wr_time = Math.round((d/wr_spd)*850).toString();
        this.skt.send(this.msg);
        this.createWorker(wr_time);
        console.log('cmmd - '+ this.msg);
    }
    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
    RControl.prototype.b_down = function() {
        if(this.speed > -300){
            this.speed = this.speed - (this.spd_inc * this.spd_factor);
        }
        else{
            this.speed = -300;
        }
        ds = this.turn();
        this.msg = 'SetMotor LWheelDist '+ Math.round(ds[0]).toString() +' RWheelDist '+
        Math.round(ds[1]).toString() + ' Speed '+ Math.round(Math.abs(this.speed)).toString();
        d = Math.min(Math.abs(Math.round(ds[0])),Math.abs(Math.round(ds[1])));
        wr_time = Math.round((d/Math.round(Math.abs(this.speed)))*850).toString();
        this.skt.send(this.msg);
        this.createWorker(wr_time);
        console.log('cmmd - '+this.msg);
    }
    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
```

```
    RControl.prototype.b_space = function() {
        this.wrkr.terminate();
        this.wrkr = null;
        msg = 'SetMotor RWheelDisable LWheelDisable';
        this.skt.send(msg);
        msg = 'SetMotor RWheelEnable LWheelEnable';
        this.skt.send(msg);
    }

    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
    RControl.prototype.turn = function() {
        arc_c = this.speed * this.time_for_a_turn; // Robot center arc displacement of robot
        center
        this.alpha = this.theta_dot * this.time_for_a_turn; // Incremental angular rotation
        while moving [rad]

        r = arc_c + (this.w * this.alpha);
        l = arc_c - (this.w * this.alpha);

        return [l,r];
    }

    // ================================================
    // ======= Gets l and r distance for movement =======
    // ================================================
    RControl.prototype.stopWRKR = function() {
        if(this.wrkr != null){
            this.wrkr.terminate();
            this.wrkr = null;
        }
    }

    return RControl;
})();
```

Anexo G

Ejemplo sencillo cliente Matlab

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\demo\demo.m miércoles, 21 de octubre de 2015 23:42

```
function demo()

%######## Global vars ######
hist = {}; % Will record all the robot data

%########## NEEDED CODE - DON'T TOUCH ######################
% UI Neato connection info
prompt = {'Connection information',};
dlg_title = 'Connection';
num_lines = 1;
def = {'172.16.10.5:20000',};  % Robot A
%def = {'172.16.10.5:20001',};  % Robot B
%def = {'172.16.10.5:20002',};  % Robot C


conn = inputdlg(prompt,dlg_title,num_lines,def);
conexion = strsplit(conn{1},':');

% Neato Connection
sck = socket(conexion{1},str2double(conexion{2}));

% If no connection, no code execution
serialOK=1;
if length(strfind(class(sck.skt()),'java.net.Socket')) == 1

    % Check if serial port is OK (required handshaking)
    data = sck.getMsg(sck);
    ok = strfind(data, 'OK');
    if length(ok) == 1
        display('Serial port (Raspberry-Neato) OK');
    else
        sck.close(sck);
        display('Serial port (Raspberry-Neato) not found, please check cable connection and
        Neato power on, and restart Raspberry');
        serialOK=0;
    end


        % Send data mode message (required handshaking)
        msg = ['DataMode listener']; % altres modes constant  moody
        data = sck.sendMsg(sck,msg);
        display(data);

        % Wait lidar start
        wait_lidar();

        %########## END NEEDED CODE ######################

        %################################################
        %### Demo robot movement and data acquisition ###

        %Read robot data & log it into "hist"
        msg = ['GetMotors LeftWheel RightWheel'];
        data = sck.sendMsg(sck,msg);
        display(data);
        hist{1,1} = char(data);
        msg = ['GetLDSScan'];
```

```
data = sck.sendMsg(sck,msg);
display(data);
hist{1,2} = char(data);

% SetMotor distance in [mm], speed in [mm/s]
% Direct movement
dist = 1000; %[mm]
speed = 120; %[mm/s]
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(dist) , '
Speed ', num2str(speed)];
datam = sck.sendMsg(sck,msg);
display(datam);

%Wait until movement stops
pause(dist/speed);

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel'];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,3} = char(data);
msg = ['GetLDSScan'];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,4} = char(data);

% 90° rigth turn
dist = 190;
speed = 120;
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(-dist) , '
Speed ', num2str(speed)];
datam = sck.sendMsg(sck,msg);
display(datam);

% Wait until movement stops
pause(dist/speed);

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel'];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,5} = char(data);
msg = ['GetLDSScan'];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,6} = char(data);

% Direct movement
dist = 1000; %[mm]
speed = 120; %[mm/s]
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(dist) , '
Speed ', num2str(speed)];
datam = sck.sendMsg(sck,msg);
display(datam);

%Wait until movement stops
pause(dist/speed);
```

```
        %Read robot data & log it into "hist"
        msg = ['GetMotors LeftWheel RightWheel'];
        data = sck.sendMsg(sck,msg);
        display(data);
        hist{1,7} = char(data);
        msg = ['GetLDSScan'];
        data = sck.sendMsg(sck,msg);
        display(data);
        hist{1,8} = char(data);

        msg = ['SetLDSRotation off'];
        data = sck.sendMsg(sck,msg);

        % Save data into workspace
        assignin('base', 'hist', hist);

        % Close connection
        sck.close(sck);
    end
end

%######## End Code ##########

%######## Functions ##########
    function wait_lidar()
        wt = waitbar(0,'Waiting Lidar Start...');
        for p=1:4
            pause(1);
            waitbar(p/4,wt,'Waiting Lidar Start......');
        end
        close(wt);
    end

end
```

Anexo H

Ejemplo sencillo cliente Python

D:\Dropbox\Dropbox\UPC\PFC\Entrega\pfc_new\Codigos\demo\demo.py miércoles, 21 de octubre de 2015 23:42

```
import socket
import sys
import time

# Global vars
rbuffer = ""

###### FUNCTIONS #######
def sendMsg(msg):
    global sock
    sock.sendall(msg + chr(10))
    return getMsg()

def getMsg():
    global sock
    global rbuffer
    resp = ''

    while not resp: # While resp empty
        rbuffer = sock.recv(8192)
        if chr(10) in rbuffer:
            resp,rbuffer = rbuffer.split(chr(10),1)
    return resp

def close():
    sock.sendall('Close' + chr(10))
    sock.close()


###### MAIN #######
if __name__ == '__main__':
    ########## NEEDED CODE - DON'T TOUCH ######################

    global sock

    # Create a TCP/IP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect the socket to the port where the server is listening
    server_address = ('localhost', 20000)
    print 'connecting to %s port %s' % server_address
    sock.connect(server_address)

    #Handshaking - Get Serial OK
    data = getMsg()
    if 'OK' in data:
        print 'Serial port (Raspberry-Neato) OK'
    else:
        sock.close()
        print 'Serial port (Raspberry-Neato) not found, please check cable connection and
        Neato power on, and restart Raspberry'
        sys.exit()

    #Handshaking - Send data mode
    print sendMsg('DataMode listener')

    #Wait lidar start
```

```
print 'Wait lidar start'
time.sleep(3) #Wait 3 seconds

########## END NEEDED CODE ######################

print sendMsg('GetMotors LeftWheel RightWheel')
print sendMsg('GetLDSScan')

#SetMotor distance in [mm], speed in [mm/s]
#Direct movement
dist = 1000; #[mm]
speed = 120; #[mm/s]
print sendMsg('SetMotor LWheelDist '+ str(dist) +' RWheelDist '+ str(dist) + ' Speed '+
str(speed))

#Wait until movement stops
time.sleep(dist/speed)

print sendMsg('GetMotors LeftWheel RightWheel')
print sendMsg('GetLDSScan')

close()
```

Bibliografía

[1] Sebastian Thrun, Wolfram Burgard, Dieter Fox. *Probabilistic robotics*. 1999-2000

[2] Peter Corke. *Robotics, Vision and Control*. ISBN:978-3-642-20143-1

[3] *Serial commands via usb connected to usb 2 serial device* [en línea]. Disponible en: https://community.particle.io/t/serial-commands-via-usb-connected-to-usb-2-serial-device/4571/16

[4] *NeatoPylot - Auto-Pilot the Neato XV-11 from Python* [en línea]. Disponible en: http://home.wlu.edu/ levys/software/neatopylot/

[5] Dave Pallot. *simple-websocket-server* [en línea]. Disponible en: https://github.com/dpallot/simple-websocket-server

[6] Mingyong Liu, Xiaokang Lei, Siqi Zhang, Bingxian Mu. *Natural Landmark Extraction in 2D Laser Data base on Local Curvature Scale for Mobile Robot Navigation*. December 14-18, 2010. Tianjin, China.