Robot auxiliar de limpieza de suelos doméstico

Ingeniería Técnica de Informática de Sistemas

Estudiante

Miguel Angel Sánchez Muñoz

Consultor

Jordi Bécares Ferrés

Fecha entrega

21 de Junio del 2014

A Sara

Resumen

Los continuos avances en robótica y en domótica están haciendo posible la irrupción de varias clases de robots domésticos. Se trata de autómatas que facilitan las tareas más rutinarias y otros orientados al puro ocio y entretenimiento, sin olvidar tareas de seguridad, atención de personas mayores, vigilancia y otros que nos depara el futuro más cercano.

Como ejemplos, entre la multitud de los existentes, tenemos el robot Roomba fabricado y vendido por iRobot o el AIBO Mind 3 de Sony, aunque la posibilidad de conexión de AIBO a un sistema de forma inalámbrica lo convierte en un domobot.

El proyecto que se quiere realizar consiste en el desarrollo y ensamblado de un robot de limpieza de suelos que se encargue de limpiar el polvo y las pelusas que se encuentren por toda la vivienda. El robot dispondría de libertad de movimiento, esquivando toda clase de obstáculos que encuentre a su paso, gracias a sensores de proximidad y de contacto directo.

Durante la ejecución de sus tareas el robot estará enviando en tiempo real los movimientos que se encuentra realizando a un servidor web a través del modulo Wifly. Si la conexión con el servidor falla por algún motivo el sistema dispone de un mecanismo de seguridad que reinicia la conexión.

Palabras clave: sistemas empotrados, sistemas embebidos, ARM Cortex M3, LPCxpresso, LPC1769, NXP, microcontroladores, sensores, wifly, domobot, domotica, Internet de las cosas.

Área de TFC: Sistemas empotrados.

Indice de contenidos

Indice de figuras

1. Introducción.

Los sistemas empotrados y las redes de sensores inalámbricas formadas por pequeños dispositivos provistas por algún tipo de sensor (temperatura, luminosidad, etc.) abarcan una amplia variedad de aplicaciones, como la domótica asistencial doméstica, control de tráfico o control ambiental, y permiten integrar funcionalidades que anteriormente eran independientes unas de otras.

1.1 Justificación.

En la época en la que nos encontramos donde el tiempo es limitado, casi siempre por motivos de trabajo, tareas cotidianas como la limpieza del hogar pasan a ser un verdadero quebradero de cabeza para muchas personas, que se ven en la necesidad de utilizar el poco tiempo libre que le queda en este tipo de tareas.

Otro punto importante, es la ayuda que estos dispositivos proporcionan a la automatización del hogar para ancianos y personas discapacitadas con movilidad reducida. Si ademas le añadimos el concepto del *Internet de las cosas,* el robot puede ser gestionado en remoto por los familiares, por ejemplo, desde cualquier parte del mundo.

En este contexto nace la idea del proyecto de un asistente doméstico autónomo que se encargue de la limpieza de los suelos de una vivienda. Dotándolo de cierta autonomía conseguimos que el usuario final no tenga que estar pendiente del dispositivo, solo siendo necesario un mínimo de mantenimiento, recambios de mopa, vaciado de contenedor de recogida de suciedad, etc.

1.2. Descripción del proyecto.

Como se ha introducido en el resumen y en el apartado anterior, el proyecto consiste crear un sistema empotrado sobre el microcontrolador Cortex-M3 LPC1769, cual controlará como mínimo la siguiente serie de funciones programadas o tareas.

- Control de motores.
- Sistema automático de control de perdida de conexión con el servidor remoto.
- Recepción y envío de datos desde Internet.

Las tareas serán programadas bajo el sistema operativo de tiempo real FreeRTOS (sistema operativo basado en Lenguaje C) siendo necesaria con anterioridad la programación de nuevas librerías y drivers para poder gestionar los nuevos componentes hardware.

Tendremos pues, un sistema autónomo en movimiento conectado a una red sin hilos que será capaz de reportar su estado al sistema de supervisión remoto y además estará dotado de un sistema de protección contra perdida de conexión.

1.3. Objetivos del TFC.

A continuación, se listarán los principales objetivos fijados para la ejecución del proyecto y una pequeña descripción de cada uno de ellos:

a) Control de motor bipolar "paso a paso".
   El sistema dispone de una librería para el control de motores paso a paso por subida de flanco. Los dos motores de los que dispone el prototipo se gestionan independientemente pudiendo alternar su giro y así permitir la realización de cambios de sentido de marcha al robot.

b) Detección de obstáculos por proximidad y detección de contactos directos.
   Para dotar al robot de una autonomía de movimientos se ha provisto de dos tipos de sensores que interaccionan con el medio que lo rodea. Principalmente el robot se

encuentra durante toda su ejecución midiendo la distancia a la que se encuentra el objeto más cercano de su frontal, si el objeto se encuentra a menos de 3cm se realiza un giro aleatorio. Si dicho objeto no se encuentra dentro del angulo de visión y el robot sufre un impacto frontal, se a provisto de un sensor de contacto que detecta la colisión e inmediatamente se realiza una pequeña marcha atrás y un giro aleatorio.

c) Envío y recepción de datos del servidor remoto.
Se ha implementado en el sistema una librería de gestión de un modulo Wifly, dotándolo de conectividad sin hilos a una red local. Utilizando este módulo el sistema enviá en tiempo real la información de los movimientos que se están llevando a cabo.

d) Control de la pérdida de conexión con el servidor.
Se ha dotado al sistema de un mecanismo de seguridad *watchdog* que provoca una reconexión a la red en el caso de perdida de comunicación con el servidor remoto. La aplicación se encuentra durante toda la ejecución consultando el estado *alive* al servidor remoto.

e) Ensamblado de todas las partes hardware en un chasis formando el robot.
Para el ensamblado del prototipo se ha utilizado como base un tablero de fibra de densidad media 3mm y el sistema motriz se ha montado con una estructura formada por piezas del sistema de construcción de modelos Meccano. Las ruedas y los engranajes de tracción también han sido provistos del sistema de construcción de modelos Meccano.

f) Sistema de debug vía puerto USB.
Se a provisto al prototipo de un sistema para posibilitar el debug de la aplicación en tiempo de ejecución haciendo uso de uno de los puertos UART internos del microcontrolador LPC1769 y el adaptador a puerto USB CP2102.

1.4. Enfoque y método seguido.

El presente proyecto está desarrollado con un enfoque didáctico, se ha llevado a cabo en diferentes ciclos, los cuales se realizan de forma secuencial si bien en todo momento se ha tenido una visión global del fin que se buscaba lo cual permite poder realizar trabajos de forma transversal cuyos resultados han sido utilizados en ciclos posteriores facilitando la solución requerida.

Así en una primera fase de búsqueda e investigación, llevada a cabo mediante las cuatro primeras pruebas de evaluación continua PECs, se ha realizado el estudio de la tecnología y herramientas requeridas tanto en el ámbito del hardware (microcontroladores, sensores, componentes electrónicos y herramientas) como en software (sistemas operativos, lenguajes y entornos de programación). Por ultimo esta fase permite la identificación de los datos a tratar y la determinación de las metodologías utilizadas para la obtención del objetivo final.

En todo momento se ha trabajado realizando pequeños códigos de prueba para medir y testar el comportamiento, por lo que ha sido muy importante trabajar ordenadamente, aplicando una nomenclatura apropiada en las funciones y variables, separando cada librería por módulo conectado o aplicación, además de comentar a nivel de código cada función, indicando los parámetros de entrada así como los parámetros de salida.

Por último la fase de documentación ha sido llevada a cabo durante toda la duración del proyecto, desde su inicio de búsqueda de documentación e información, como la realización de documentos tales como diagramas de bloques, diagramas de flujos y esquemas eléctricos, que se aportan en esta memoria.

Se ha tenido presente la utilización de código fuente de terceros, así el proyecto se beneficia de librerías de código de otros autores el cual ha sido convenientemente modificado conseguir nuestros objetivos. El proyecto se presenta como Open Source dotando de una bidireccion al código, posibilitando igualmente que otros autores los utilicen y modifiquen en sus proyectos.

1.5. Planificación del proyecto.

La planificación inicial se hizo dividiendo el proyecto en una serie de tareas. Así se pretendía poder hacer implementaciones parciales que facilitaran hacer un seguimiento preciso del desarrollo del proyecto. Estas implementaciones parciales también deberían permitir hacer pruebas parciales del producto para poder detectar lo antes posible cualquier error.

La planificación inicial puede verse en el siguiente diagrama de Gantt:

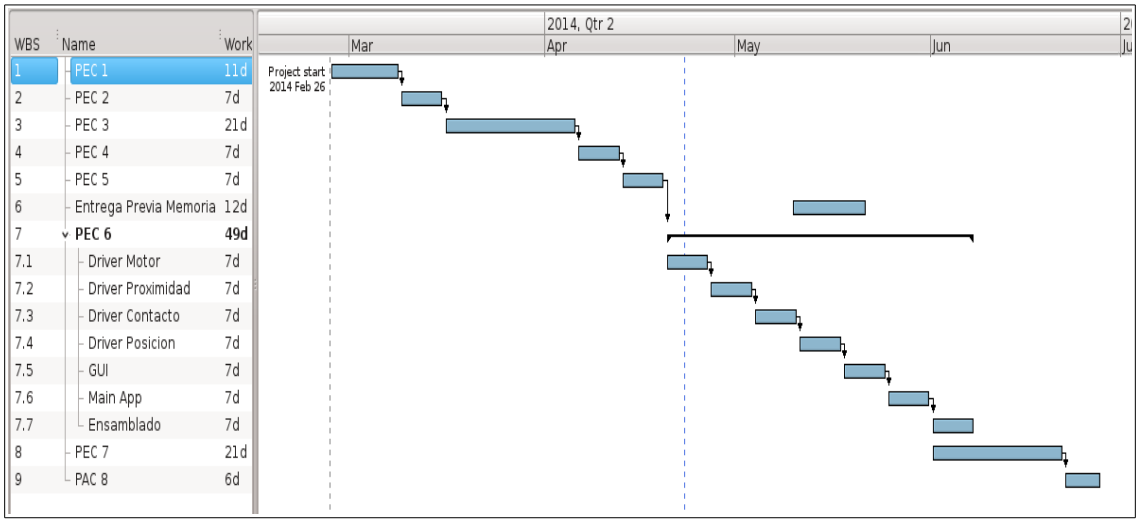

Ilustración 1: Planificación inicial

Aunque desde el principio se ha intentado seguir este diagrama de forma estricta, lo cierto es que no ha sido posible. Se han producido algunas desviaciones que han provocado un reordenamiento y eliminación de algunas de las tareas. A pesar de ello, se ha intentado en todo lo posible respetar las fechas de entrega.

Finalmente puede verse como queda la planificación en el siguiente diagrama de Gantt:

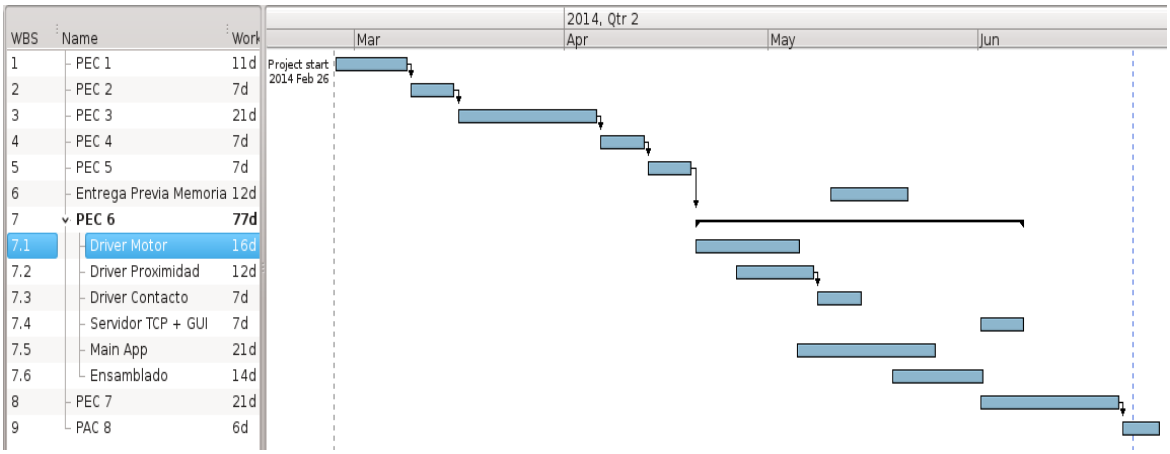

Ilustración 2: Planificación final

Las diferencias entre la planificación inicial y la final se han debido principalmente al motivo de encontrar la documentación necesaria para llevar a cabo los requisitos antes del tiempo previsto. La tarea planificada como “Driver posición” no se ha podido finalizar, siendo eliminada de la planificación y no entregando la parte de código que la implementaba.

1.6. Recursos empleados.

Para la realización del proyecto se han empleado una gran variedad de recursos que se detallan a continuación.

Recursos hardware

- LPC1769 LPCXpresso.

  Placa principal del sistema formada por la placa del microcontrolador Cortex-M3 combinada con un programador/depurador o JTAG. Incluye, entre otros, 64kB SRAM, 512Kb Flash, 4 UART, 3 I2C, SPI, ADC de 12 bits de resolución y 8 canales, DAC de 10 bits, timers de propósito general y 70 GPIO.

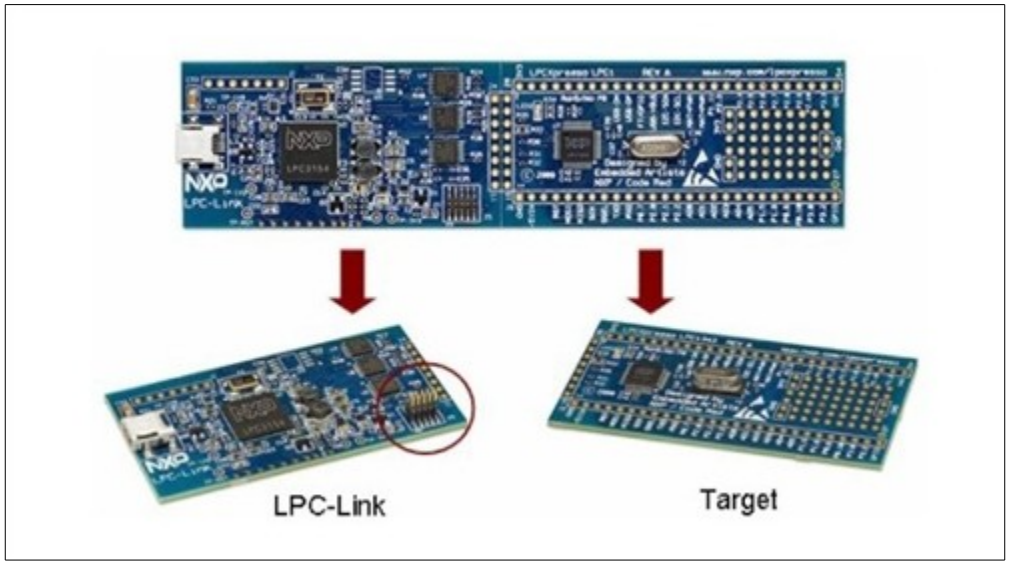

Ilustración 3: LPC1769 LPCXpresso board

- Motor NEMA 17.

  Este motor paso a paso NEMA 17 es bipolar, tiene un ángulo de paso de 1.8º (200 pasos por vuelta) y cada bobinado es de 1.2 A a 4v, capaz de cargar con 3.2 kg/cm (44 oz-in).

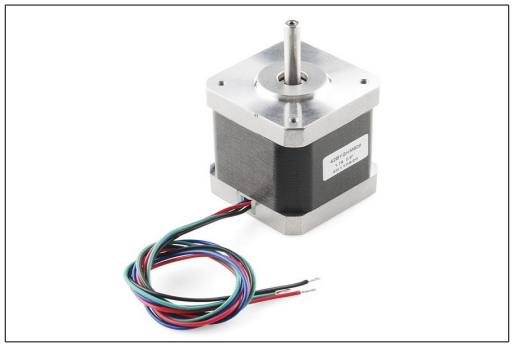

Ilustración 4: Motor NEMA 17

- Pololu A4988 steptick.

  Esta placa de Pololu utiliza el driver Allegro A4988 bipolar para motores paso a paso. Este driver tiene limitación de corriente ajustable, protección contra sobre corriente y cinco

resoluciones diferentes de microstepping. Funciona desde 8V a 35V y puede suministrar 1A por bobina sin usar ventilación forzada o un disipador.

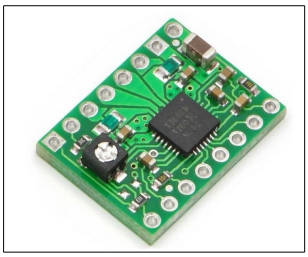

Ilustración 5: Pololu A4988 steptick

- Sensor de proximidad por ultrasonidos HCSR04.

  Éste sensor PING funciona como un sonar mediante ultrasonidos y es capaz de detectar objetos a una distancia de entre 2 centímetro a 3 metros. En su pin de salida podremos medir el ancho de pulso PWM en función de la distancia del obstáculo.

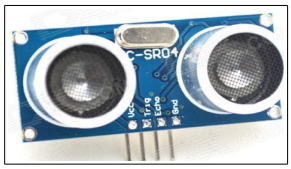

Ilustración 6: Sensor de proximidad HCSR04

- AVR ATtiny85.

  Microcontrolador de 8 bits fabricado por Atmel.

  Características:

  - Memoria ROM: 8Kb.
  - Memoria RAM: 512 bytes.
  - Pines I/O: 6.
  - Frecuencia: 20 Mhz con cristal externo.

- Permite programación ICSP.

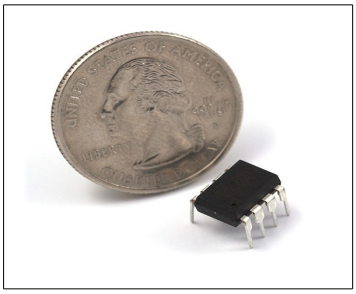

Ilustración 7: AVR ATtiny85

- Micro-interruptor.

  Micro-interruptor de propósito general con palanca utilizado como detector de contacto directo. Soporta hasta 5A a 250VAC.

Ilustración 8: Micro-interruptor

- RN-XV WiFly 802.11 b/g.

  Está basado en el robusto módulo RN-171 de Roving Networks e incorpora un emisor 802.11 b/g, un procesador interno de 32bits, pila TCP/IP, RTC, crypto, gestión de energía e interfaz analógica para sensores externos. Viene precargado con el firmware original de Roving Networks para simplificar la integración y minimizar el tiempo de desarrollo. En la configuración más simple, tan sólo necesita de 4 pines para funcionar (PWR, TX, RX y GND) y crear una conexión inalámbrica Wifi.

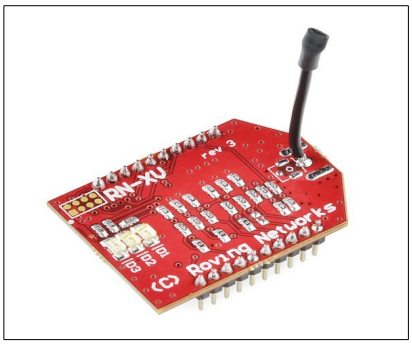

Ilustración 9: RN-XV WiFly 802.11 b/g

- CP2102

  Adaptador USB-serie con terminales TX, RX, 3v3, 5v y GND. Crea un puerto virtual COM para conectar el microcontrolador a un terminal de comunicaciones a través de un puerto USB.

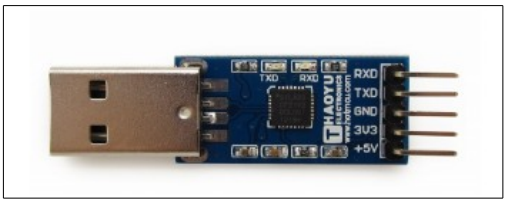

Ilustración 10: CP2102

- Batería.

  Batería recargable de Li-ion, 12V, 6800mAh.

Ilustración 11: Batería recargable

- Componentes varios.

  Protoboard, resistencias, diodo led y reguladores de tensión 3v3 y 5V,

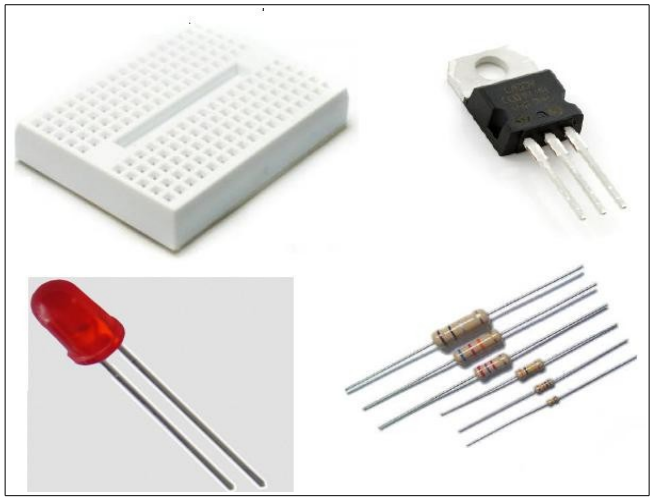

Ilustración 12: Componentes varios

Recursos software

- Entorno de desarrollo LPCXpresso IDE v7.0.2.
- Sistema operativo en tiempo real FreeRTOS v7.1.0.
- Librerías de desarrollo CMSISv2p00_LPC17xx.
- Arduino IDE v1.0.5.
- Lenguaje de programación C.
- Apache HTTP Server + PHP.

1.7. Productos obtenidos.

Tras la conclusión del proyecto se han obtenido los siguientes productos:

- Prototipo de robot con todas las partes hardware ensambladas y testadas.

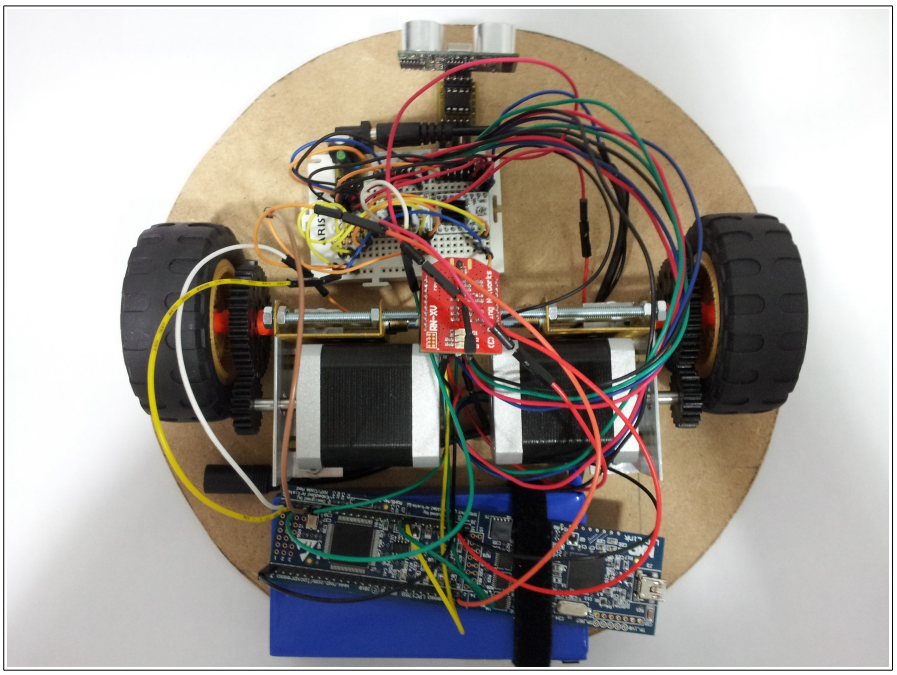

Ilustración 13: Prototipo ensamblado

- Proyecto ejecutable “TFC_Robotclean” que constituye el código principal de la aplicación que controla el robot. Esta desarrollado en lenguaje C sobre el sistema operativo en tiempo real FreeRTOS. Utilizando las librerías ARM® Cortex® Microcontroller Software Interface Standard para el microcontrolador LPC1769.
- Proyecto de librería estática “TFC_Library” que implementa las funciones para acceso a periféricos y de apoyo a la aplicación. Estas librerías se explicarán con más detalles en los siguientes apartados.
- Servidor web desarrollado en lenguaje PHP sobre Apache HTTP Server, que implementa un puerto de escucha TCP y una interfaz simple de usuario.

1.8. Breve descripción de los otros capítulos de la memoria.

En el siguiente capítulo haremos una pequeña introducción al mundo de los sistemas empotrados, la actualidad y el presente de los mismos, así como pequeño estudio de mercado sobre nuestro sistema. Posteriormente en el capítulo 3 describiremos a grandes rasgos el funcionamiento y diseño global de nuestro sistema.

Entrados en el capítulo 4, nos adentraremos de una forma más técnica y detallada en el funcionamiento y diseño del proyecto. Ya en el capítulo 5, expondremos un estudio de la viabilidad técnica del sistema, para posteriormente en el capítulo 6, exponer una valoración económica orientativa del coste del sistema.

Por último, en el capítulo 7 expondremos las conclusiones extraídas de nuestro trabajo, exponiendo propuestas de mejoras y realizando una autoevaluación de los objetivos propuestos en el capitulo 1.

2. Antecedentes.

2.1. Estado del arte.

Un sistema empotrado se puede definir como un sistema dedicado, diseñado exclusivamente para la realización de unas tareas especificas y que frecuentemente es un sistema de computación en tiempo real, por lo que son desarrollados para la realización de tareas muy concretas y que por lo general requieren de una respuesta en tiempo real.

Están compuestos por una CPU ó MCU, memoria , buses de datos y puertos de entrada y salida. Suelen incorporar en su estructura dispositivos conversores analógico-digitales (ADC), digital-analógico (DAC), moduladores de pulsos (PWM), y diversos interfaces como pueden ser Ethernet, UART, I2C, SSP, CAN, que le permiten toda una interconexión con el mundo real y poder interaccionar con dispositivos periféricos como pueden ser sensores, motores, redes, u otros sistemas computacionales de mayor envergadura.

Las características del sistema embebido principalmente están determinadas por el tipo de CPU ó MCU integrada, siendo las principales de estas características: La velocidad de procesamiento, longitud de palabra MCU, cantidad de memoria RAM ó Flash y número de dispositivos interfaces.

En la actualidad existen gran diversidad de CPU.s, MCU.s en el mercado en la siguiente tabla vemos algunas de las más utilizadas y sus características principales.

	CPU	Longitud De Palabra	RAM	ROM	Frecuencia	Periféricos	Coste
LPC1769	ARM Cortex M3	32 bit	64 Kb SRAM	512Kb Flash	120 Mhz	UART: 4 SPI: 4 I2C: 3 ADC: 8 PWM: 6 USB: 1 Etherner:1	13,61 €
ATMega1280	MCU AVR	8 bit	8 Kb	128Kb Flash	16 Mhz	USART: 4 SPI: 1 I2C: 1 ADC: 16	7,52 €
PIC18F8720	RISC	8 bit	3 Kb SRAM	128Kb Flash	25 Mhz	UART: 2 SPI: 1 I2C: 1 ADC: 16 PWM: 5	9,86 €
MK11DX256VMC5	ARM Cortex M4	32 bit	32 Kb	256Kb Flash	50 Mhz	USART: 4 SPI: 2 I2C: 1 ADC: 16	14,04 €
MSP430F5438	MPS Risc	16 bit	32 Kb	512Kb Flash	25 Mhz	USCI A: 4 USCI B: 4 ADC: 16	10,75 €

Ilustración 14: Tabla comparativa CPU

Como puede verse existe una extensa oferta disponible. En este proyecto la CPU se nos aporta en la asignatura por lo que no hay razón para justificar la elección, aunque si cabe indicar que se considera una buena elección pues esta se encuentra entre las mas avanzadas ofreciendo grandes posibilidades.

A esta selección de CPU, el sistema tiene que unir las red de sensores sin hilos o WSN (Wireles Sensor Network) la cual esta compuesta, de forma genérica por un conjunto de sensores distribuidos. Cada sensor consta de un sistema empotrado y de la sensorización específica necesaria para cada proyecto.

Así la sensorización puede ser parte del sistema embebido o puede acoplarse a través de las correspondientes interfaces analógicas o digitales, siendo en este caso tratadas por el sistema embebido como periféricos.

Visto este breve resumen de los componentes de Hardware que conforma un sistema embebido, cabe indicar que este hardware tiene fuertes limitaciones como se ha visto, alguna de estas son tamaño de memoria, capacidad de computo, comunicación con su entorno, consumo de energía etc. de igual forma debe adaptarse a condiciones externas debidas a ubicación, movilidad, condiciones de trabajo, que lo hacen especial. Debido a estas diferencias los Sistemas Operativos para este tipo de hardware presentan también diferencias con respecto a los sistemas convencionales.

2.2. Estudio de mercado.

En la actualidad existen muchos fabricantes que fabrican y comercializan productos como el desarrollado en esta memoria. Mostramos dos ejemplos de robots que se encuentran en venta en la actualidad:

- **ViRobi** es un robot mopa autónomo. Recoge polvo, pelos y pelusas navegando de forma autónoma. De forma circular**,** tiene un diámetro de algo más de 29 centímetros y 6,3 cm de altura, lo que le permite introducirse por debajo de algunos muebles para limpiar. Su peso es de 4,4 kilos**.**
-

Ilustración 15: ViRobi Robot

- **Roomba** es un aspirador robótico fabricado y vendido por iRobot. Se comercializa como *iRobot Roomba Robotic Floorvac.* El Roomba se lanzó al mercado en 2002. En 2004 comenzó a venderse la segunda generación de modelos, en 2007 la tercera y en 2010 la quinta. Forma circular de 34 cm de diámetro y menos de 9 cm de altura. Cuenta con sensores de contacto con paredes y muebles e infrarrojos. A partir de la segunda generación las unidades cuentan con un sensor de suciedad que les permite centrarse en los puntos más sucios. Si los sensores detectan que la unidad queda atascada, ha sido levantada o no puede salir de una zona estrecha, una alarma comienza a sonar para que el usuario pueda localizar la unidad.

Ilustración 16: Roomba

Tabla comparativa de precios de productos de distintos fabricantes:

	**iRobot Roomba 655**	**429€**
	**LG VR5906LV**	**299€**
	**Samsung VCR8980L3K**	**649€**
	**Vileda ViRobi**	**44,95€**
	**Vileda 137172**	**179€**

Ilustración 17: Comparativa de precios

Una vez visto el mercado podemos sacar como conclusión que el principal inconveniente que se le puede achacar a la adquisición de uno de estos dispositivos, es el elevado coste que tienen, si buscamos dispositivos con algunas características medianamente avanzadas.

En nuestro caso el robot desarrollado en este proyecto se encuentra a medio camino entre el ViRobi Robot mopa de Vileda y el Roomba, ademas añadiéndole la conectividad del *Internet de las cosas,* el sistema de conectividad no le implementan prácticamente ningún fabricante para este tipo de dispositivo. Esto es un punto a favor de nuestro prototipo siendo una ampliación al mercado actual.

3. Descripción funcional.

Con el objetivo de tener una visión global, comenzaremos exponiendo la funcionalidad del sistema al completo, de una forma gradual y progresiva.

3.1. Robot limpiador.

Primero podemos realizar una descripción del sistema físico que componen todas las partes de robot:

Se ha dispuesto de una plataforma circular que alberga todos los componentes que forman el robot. Sobre su eje central se han instalado los motores que junto al sistema de engranajes tractores y las dos ruedas forman el conjunto de sistema tractor. Esta configuración permite mayor facilidad de movimiento a todo el conjunto. Sobre dicha plataforma también se encuentra instalado toda la parte electrónica junto a la batería que ofrece una autonomía de movimiento total sin ninguna conexión eléctrica con ningún elemento externo al conjunto.

En la ilustración 13 podemos ver una fotografiá que muestra todo el sistema físico ensamblado, pudiendo reconocer cada parte que forma el robot.

El funcionamiento del sistema una vez puesto en marcha se describe en los puntos siguientes:

- En el momento de arranque, el robot haciendo uso de su autonomía, comienza con sus motores en una configuración de desplazamiento hacia delante.

- A la par de encontrarse ya en movimiento, el robot se conectará a la red sin hilos conilustraciónda. Una vez que establece conexión con el servidor remoto el robot comienza a enviar los datos de los movimientos que se encuentra realizando y a su vez se inicia el sistema de seguridad watchdog.

- Si durante su ejecución de movimiento el robot se encuentra a menos de 3 cm de cualquier obstáculo, el robot realiza un movimiento de giro sobre su eje central aleatorio, derecha o izquierda, de 45º y prosigue su marcha adelante.

- Ademas del punto anterior, si durante la ejecución de movimiento el robot no detecta el obstáculo y sufre una colisión, el sistema realiza dos operativas de movimiento, un movimiento de marcha atrás para retirarse levemente del obstáculo en cuestión con el que ha sufrido la colisión y realiza un movimiento de giro sobre su eje central aleatorio, derecha o izquierda, de 45º y prosigue su marcha adelante.

- Durante toda la ejecución de la actividad de movimientos del robot, si este en algún punto pierde la conexión con el servidor remoto, el sistema de seguridad watchdog iniciado durante la conexión inicial a la red, realiza un reinicio de dicha conexión, reconectando de nuevo a la red conilustraciónda y al servidor remoto.

- El sistema dispone de un led en función de parpadeo que indica que el sistema se encuentra operativo y realizando un funcionamiento correcto.

En la siguiente ilustración, podemos ver el diagrama de bloques del sistema total con todas las partes que lo componen:

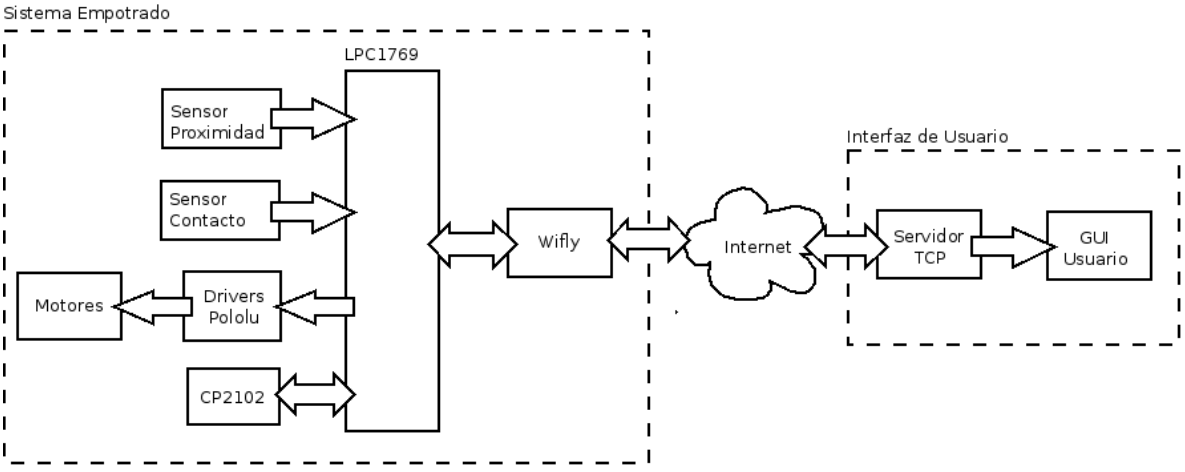

Ilustración 18: Diagrama de bloques. Sistema Total

3.2. Aplicación web.

Se ha decidido crear una interfaz básica para la supervisión del robot, ya que el diseño y programación de una aplicación web, conlleva muchas horas de trabajo y va más allá del alcance de este proyecto.

La interfaz web esta formada por dos partes, un servidor con un puerto TCP a la escucha y una GUI simple para la visualización por parte del usuario. Se ha diseñando una GUI portable e instalable en básicamente cualquier servidor web que cumpla ciertos requisitos, se detallará estos requisitos y otros detalles técnicos en el siguiente capítulo.

En la siguiente ilustración, podemos ver el diagrama de bloques de la GUI donde se aprecian las dos partes que la componen y como interaccionan con el prototipo:

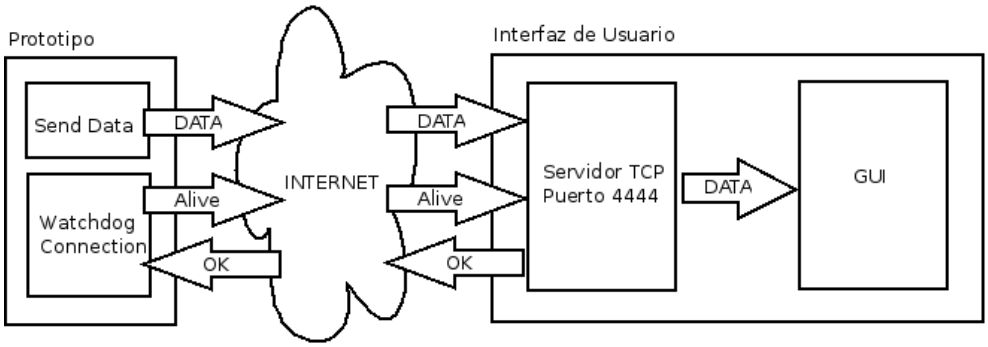

Ilustración 19: Diagrama de bloques Interfaz de usuario

3.3. Aplicación Principal.

Como se ha comentado en capítulos anteriores, nuestro sistema empotrado tendrá alojada una aplicación desarrollada bajo el sistema operativo en tiempo real FreeRTOS. Se ha optado por desarrollar una aplicación que tiene su base en aplicaciones desarrolladas y estudiadas en la fase

de formación, es decir, hemos desarrollado una aplicación con el objetivo de tener un sistema productor/consumidor.

La aplicación consiste en un proyecto principal, llamado *TFC_Robotclean*, y un proyecto de librería estática llamado *TFC_Library*. Esta librería se utilizan básicamente para acceder a los diferentes periféricos, y por implementar diversas utilidades que serán llamadas desde diferentes puntos de la aplicación.

A continuación haremos una descripción básica de los archivos de los que consta el proyecto *TFC_Robotclean*:

1. **main.c**: Es la función principal de la aplicación, consta de 7 tareas que realizan las siguientes funciones:

    - vTaskMotorDefault. Tarea que se encarga de conilustraciónr los motores en su posición por defecto, en esta posición los motores realizan un movimiento hacia adelante.
    - VtaskMotorGiro. Tarea que se encarga de conilustraciónr los motores en posición de giro, el sentido de giro se genera aleatoriamente y se ejecuta realizando un giro de 45º.
    - VtaskMotorBack. Tarea que se encarga de conilustraciónr los motores en posición de marcha atrás y realizar dicho movimiento.
    - VtaskConnection. Tarea que se encarga de realizar la conexión con la red sin hilos y con el servidor remoto TCP.
    - VtaskWatchDogConnection. Tarea que forma el sistema de seguridad watchdog, es la encargada de comprobar el estado de la conexión con el servidor remoto y el reinicio de la conexión si esta se pierde.
    - VtaskSendData. Tarea encargada del envió de los datos al servidor remoto.
    - VtaskRun. Tarea que muestra el estado del funcionamiento correcto del sistema mediante un led parpadeante.

A continuación haremos una descripción básica de las librerías que se implementan dentro del proyecto de librería estática *TFC_Library* y se utilizan en el desarrollo de este proyecto:

1. gpio.h: Librería que implementa la funciones necesarias para la utilización de las salidas digitales, ademas de las interrupciones externas GPIO.
2. motor.h: Librería que implementa la funciones necesarias para la utilización de motores paso a paso junto a un driver Pololu A4988.
3. wifly.h: Librería que implementa la funciones necesarias para la utilización y comunicación mediante un modulo Wifly.
4. debug.h: Librería que implementa la funciones necesarias para la realización de debugger de la aplicación, utilizando como salida un puerto UART en el que se ha conectado el módulo conversor de serie a USB CP2102.
5. led.h: Librería que implementa la funciones necesarias para el uso de un diodo led.

En la siguiente ilustración, podemos ver el diagrama de bloques de la aplicación:

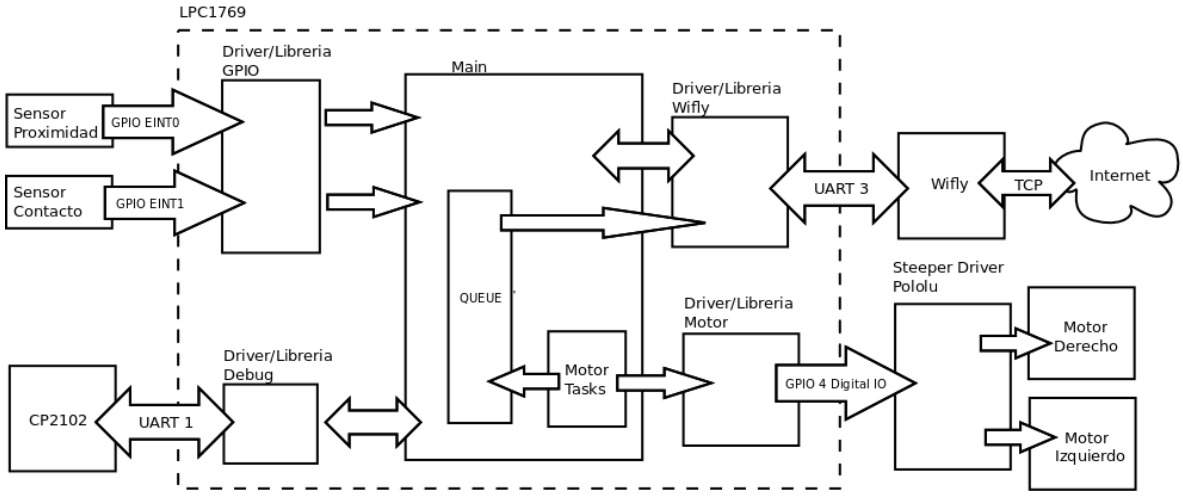

Ilustración 20: Diagrama de bloques aplicación "TFC_Robotclean"

Como se aprecia en la imagen anterior, por un lado vemos en el bloque central la aplicación principal, sustentándose por unas librerías o drivers diseñados para cada dispositivo hardware.

Esta solución de diseño nos ha permitido desarrollar una aplicación de forma limpia, siendo más fácil la localización de errores y permitiéndonos en el futuro poder utilizar estas mismas librerías en otras aplicaciones para otros sistemas (portabilidad e independencia).

Como se ha comentado en este capítulo, en el siguiente capítulo nos adentraremos en detalles más técnicos de la aplicación del sistema.

4. Descripción detallada.

En este apartado se van a ampliar los diferentes apartados del punto 3 de forma técnica y detallada, incluyendo diagramas de ejecución, casos de uso y diagramas de flujo.

4.1 Hardware y módulos funcionales.

Una vez que conocemos el funcionamiento básico de cada objeto hardware en el proyecto y a que módulo funcional pertenece, nos adentraremos en las siguientes secciones de una forma más técnica con cada uno de ellos, observando su integración progresiva en el sistema total.

4.1.1 Módulo Sistema principal LPC1769.

El microcontrolador LPC1769 como se ha comentado con anterioridad, está basado en un Cortex-M3 de ARM y principalmente ha sido concebida para el desarrollo de aplicaciones embebidas con un alto nivel de integración y con un bajo nivel de consumo.

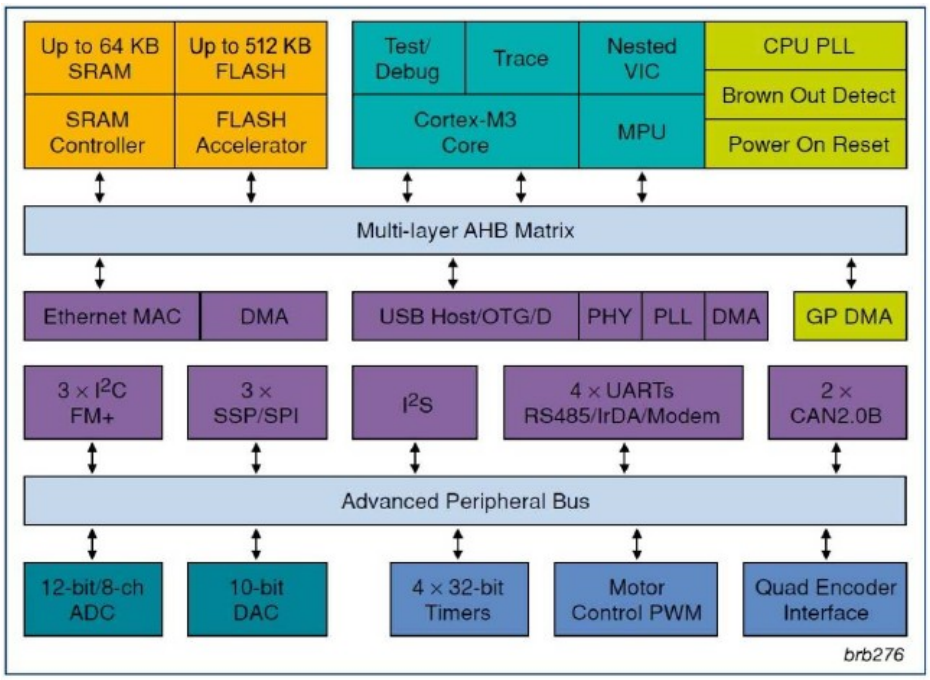

Ilustración 21: Diagrama de bloques LPC1769

Algunas de sus principales características son:

- Frecuencia de trabajo de 120 Mhz.
- 512 Kb de memoria flash.
- 64 Kb de memoria de datos.
- Puerto Ethernet.
- Interfaz USB Device/Host/OTG.
- 8 canales DMA.
- 4 puertos UART.
- 2 canales CAN.
- 2 controladores SSP.
- 1 interfaz SPI.
- 3 interfaces I2C.
- 6 interfaces PWM.

Y muchas otras características más que se pueden consultar en la página oficial del fabricante (consultar capítulo 10 Biografía).

Antes de la conexión de cualquier hardware en la mota, se ha procedido a al estudio de la misma en la fase de formación.

El primer paso ha sido familiarizarse con el IDE suministrado (basado en el entorno de desarrollo “Eclipse”), para posteriormente ir testando proyectos demostrativos facilitados por el mismo fabricante. Con estos ejemplos se obtienen las primeras nociones de sistema operativo FreeRTOS así como del hardware.

Además, el fabricante pone a disposición una carpeta con varios proyectos de ejemplo demostrativos y comentados del IDE con la LPC1769, ubicados en la carpeta “Examples” del propio IDE de desarrollo, utilizando para ello librerías ya implementadas por el fabricante para cada tipo de comunicación.

Una vez formados y familiarizados con el sistema de desarrollo y el microcontrolador, se procedido a crear por partes cada librería y así poder ir testando por partes los módulos funcionales que instalaremos. En el apartado 4.3 se entrará en más detalle sobre la “aplicación principal final”.

4.1.2 Módulo Control de motores.

Ya que se quiere dotar de una mayor precisión en el control de giro de los motores se ha optado por la utilización de motores del tipo paso a paso bipolar, en concreto NEMA 17. La corriente de funcionamiento de estos motores es de 12v y para el control se han instalado dos Pololu A4988 steptick. Esta placa utiliza el driver Allegro A4988 bipolar para motores paso a paso. Este driver tiene limitación de corriente ajustable, protección contra sobre corriente y cinco resoluciones diferentes de microstepping. Funciona desde 8V a 35V y puede suministrar 1A por bobina sin usar ventilación forzada o un disipador. En la siguiente ilustración se muestra el pineado del que dispone este driver:

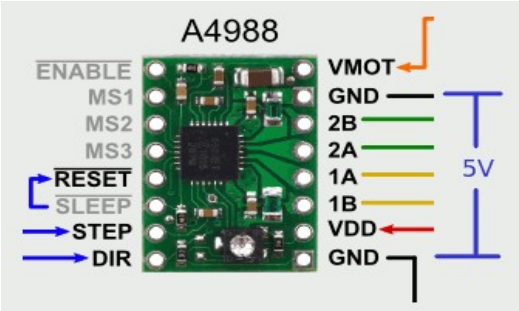

Ilustración 22: Pineado Pololu A4988

Ilustración que muestra el conexionado eléctrico entre el microcontrolador LPC1769 y los drivers A4988:

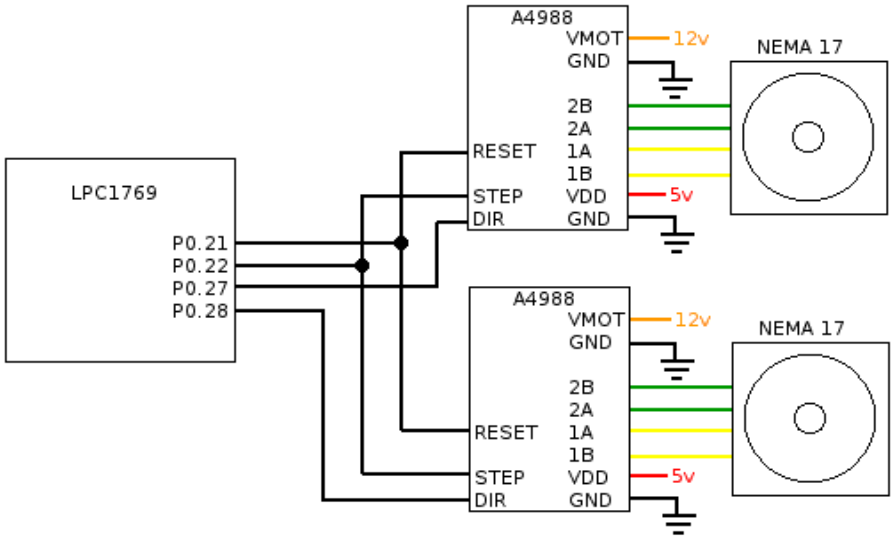

Ilustración 23: Esquema eléctrico de conexión del LPC1769 con los drivers A4988

Una vez realizadas las conexiones se procedió al desarrollo de una librería llamada motor.h, la cual implementa las siguientes funciones utilizadas en nuestro proyecto:

1. Motor_Init().

   Función encargada de inicializar todos los pines que forman el control de los motores.

2. Motor_Steep().
   Función que genera un pulso de subida “steep” en el pin pasado como parámetro, el motor asignado a ese pin se moverá un paso.
3. Motor_Start().
   Función que habilita el movimiento de los motores.
4. Motor_Stop().
   Función que detiene el movimiento de los motores.
5. Motor_Go().
   Función que conilustración los pines de dirección para que el robot se mueva hacia adelante.
6. Motor_Back().
   Función que conilustración los pines de dirección para que el robot se mueva hacia atrás.
7. Motor_DirRight().
   Función que conilustración los pines de dirección para que el robot realice un giro hacia la derecha.
8. Motor_DirLeft().
   Función que conilustración los pines de dirección para que el robot realice un giro hacia la izquierda.
9. Motor_DirRandom().
   Función que genera una aleatoriedad en el sentido de giro. Una vez generado utiliza la funciones descritas anteriormente para conilustraciónr el sentido de giro.

4.1.2 Módulo Sensores de proximidad y contacto directo.

El prototipo esta dotado de dos tipos de sensores, de proximidad por ultrasonidos HCSR04 y de contacto directo utilizando un microinterruptor.

El sensor de proximidad por ultrasonidos HCSR04 funciona midiendo el tiempo de “echo” que transcurre desde el envío de la señal “trig”. Esta función se ha implementado sobre un microcontrolador ATtiny85 de Atmel utilizando el entorno de desarrollo Arduino. De esta manera se ha simplificado el desarrollo de la aplicación principal sobre el LPC1769.

Una vez iniciado nuestro sistema el sensor de proximidad esta continuamente midiendo la distancia a la que se encuentra el obstáculo que se encuentre enfrente de él. Si dicha distancia es inferior a 3cm el ATtiny85 genera un pulso en su pin 6 que activa la interrupción EINT0 en el pin

P2.10 del LPC1769. En el programa principal de nuestro sistema se ha implementado la función EINT0_IRQHandler que inicia la tarea vTaskMotorGiro encargada de realizar el giro del robot.

El sensor de contacto directo está formado por un microinterruptor conectado directamente al pin P2.11 del LPC1769. Este pin está conilustracióndo como interrupción externa. Al accionar el microinterruptor por una colisión se activa la interrupción EINT1. En el programa principal de nuestro sistema se ha implementado la función EINT1_IRQHandler que inicia la tarea vTaskMotorGBack encargada de realizar una ligera marcha atrás del robot.

En la siguiente ilustración podemos observar el esquema eléctrico de conexión de los dos sensores con el LPC1769:

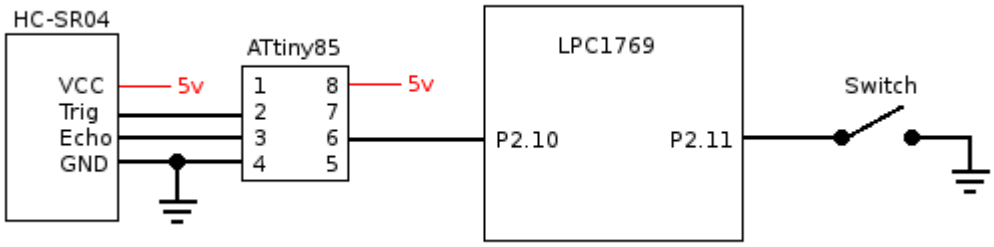

Ilustración 24: Esquema eléctrico de conexión del LPC1769 con los sensores de proximidad y contacto

4.1.3 Módulo Wifly.

El módulo WyFly RN-XV es una solución ideal para dotar a nuestro sistema LPC1769 de conectividad sin hilos 802.11 b/g. Este dispositivo en sí, ya es un sistema empotrado, ya que está dotado por un microcontrolador de 32 bits de SPARC, 8 Mb de memoria flash, 128 Kb de memoria RAM, ultra bajo consumo, etc..

El modelo utilizado en el desarrollo de este proyecto es el RN-171, el cual dispone de antena incorporada y el siguiente consumo en lo que ha transmisión se refiere conilustraciónble por el usuario 4uA sleep, 35mA Rx, 185 mA Tx at 12dBm.

Como veremos en las siguientes ilustraciones, el sistema empotrado que lo forma no solamente dispone de un puerto UART, si no también de funciones y puertos adicionales como E/S GPIOs, ADCs, etc..

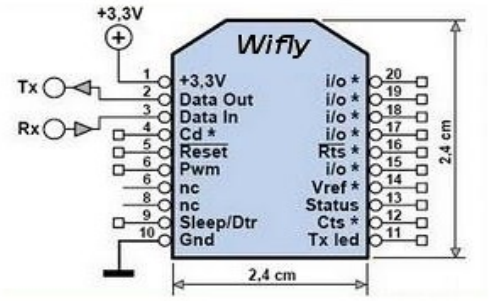

Ilustración 25: Pineado modulo Wifly

Pad Number	Signal Name	Description	Optional Function	Direction
1	VDD_3V3	3.3V regulated power input to the module		POWER
2	UART_TX	UART TX, 8mA drive, 3.3V tolerant		OUT →
3	UART_RX	UART RX, 3.3V tolerant		IN ←
4	GPIO 8	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
5	RESET	**Optional** Module Reset Signal (active low), 100k Pull up, apply pulse of at least 160us, 3.3V Tolerant		INPUT
6	GPIO 5	GPIO, 24mA drive, 3.3V tolerant	Data TX/RX	OUT
7	GPIO 7	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
8	GPIO 9	Enable Adhoc mode, Restore factory defaults, 8mA drive, 3.3V tolerant		IN / OUT
9	GPIO 1	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
10	GND	Ground		GND
11	GPIO 14	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
12	UART_RTS	UART RTS flow control, 8mA drive, 3.3V tolerant		OUT →
13	GPIO 4/SEN 6	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant).** Defaults to GPIO 4	Association Status	IN / OUT
14	Not Used			No Connect
15	GPIO 6/SEN 7	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant).** Defaults to GPIO 6	Connection Status	POWER
16	UART_CTS	UART CTS flow control, 3.3V tolerant		IN ←
17	SENSOR 5	Sensor Interface, Analog input to module, (**3.3V tolerant)**		INPUT
18	GPIO 3/SEN 4	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to GPIO 3		IN / OUT
19	GPIO 2/SEN 3	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to SEN 3		IN / OUT
20	SEN 2	Sensor Interface, Analog input to module, **3.3V tolerant**		INPUT

Ilustración 26: Funciones del modulo Wifly

En la siguiente ilustración podemos observar el esquema eléctrico de conexión del módulo Wifly con el LPC1769:

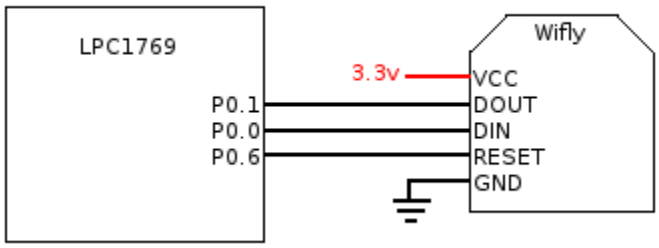

Ilustración 27: Esquema eléctrico de conexión del LPC1769 con el modulo Wifly

Una vez realizadas las conexiones se procedió al desarrollo de una librería llamada wifly.h, la cual implementa las siguientes funciones utilizadas en nuestro proyecto:

1. Wifly_INIT().
   Función que inicializa el modulo Wifly. Recibe como parámetros el puerto UART y el baudrate. Conilustración el baudrate del puerto UART en el que se ha conectado el módulo.
2. Wifly_WlanConnection().
   Función que realiza la conexión con la red inalámbrica. Recibe como parámetros el SSID de la red, el modo de autenticación y la clave de acceso a la red.
3. Wifly_OpenHost().
   Función que establece una conexión TCP con un servidor remoto. Recibe como parámetros la dirección del host remoto, el número de puerto que se encuentra a la escucha en el host remoto y el tiempo de reconexión.
4. Wifly_SendDataHost().
   Función que realiza el envío de datos al servidor remoto. Recibe como parámetro los datos que se quieren enviar.
5. Wifly_ReceiveData().
   Función que recibe los datos del servidor remoto. Recibe como parámetro el tiempo que se va a estar recibiendo como timeout.

4.1.3 Módulo Debug.

Haciendo uso del modulo conversor serie/USB CP2102, se ha desarrollado un sistema de debug para poder visualizar en tiempo real, a través de un puerto UART del LPC1769, la ejecución de la aplicación principal.

En la siguiente ilustración se muestra el pineado del que dispone el modulo CP2102:

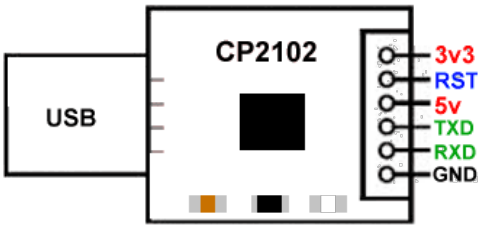

Ilustración 28: Pineado modulo CP2102

En la siguiente ilustración podemos observar el esquema eléctrico de conexión del módulo CP2102 con el LPC1769:

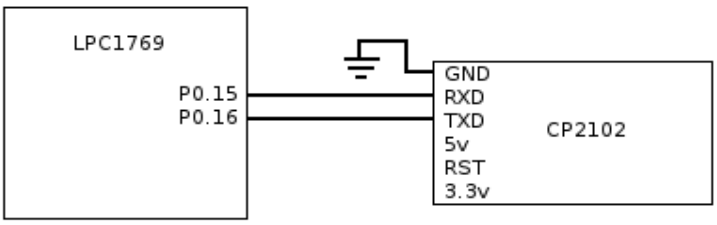

Ilustración 29: Esquema eléctrico de conexión del LPC1769 con el modulo CP2102

Una vez realizadas las conexiones se procedió al desarrollo de una librería llamada debug.h, la cual implementa las siguientes funciones utilizadas en nuestro proyecto:

1. debug_INIT().
   Función que inicializa el modulo CP2102. Recibe como parámetros el puerto UART y el baudrate. Conilustración el baudrate del puerto UART en el que se ha conectado el módulo.
2. debug().
   Función que imprime por el puerto UART conilustracióndo como debug los datos recibidos como parámetro.

4.1.3 Módulo Run.

Este módulo consta de un diodo led conectado a un pin del LPC1769 en modo parpadeo, que indica de forma visual el correcto funcionamiento del sistema.

En la siguiente ilustración podemos observar el esquema eléctrico de conexión del diodo led con el LPC1769:

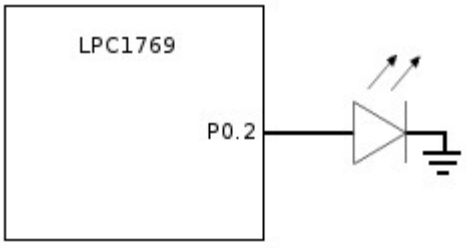

Ilustración 30: Esquema eléctrico de conexión del LPC1769 con el led.

Una vez realizadas las conexiones se procedió al desarrollo de una librería llamada led.h, la cual implementa las siguientes funciones utilizadas en nuestro proyecto:

1. led_init().
   Función que inicializa el pin al que se encuentra conectado el led. Recibe como parámetro el pin digital.
2. led_toggle().
   Función que realiza un cambio de estado en el pin al que se encuentra conectado el led.

4.2. Aplicación web.

Como se ha introducido anteriormente, la aplicación web o interfaz de usuario esta compuesta por dos partes, un servidor con un puerto a la escucha TCP y una interfaz de usuario web.

4.2.1 Servidor TCP.

El servidor TCP se ha implementado utilizando el lenguaje de programación PHP. Este está formado por un script que se ejecuta en el host remoto. Este script crea un socket en el host remoto sobre el puerto 4444 y se mantiene a la escucha.

En la siguiente ilustración se muestra la salida de consola del comando netstat, en la que se puede comprobar que el servidor se encuentra a la escucha en el puerto 444:

```
[miguel@server ~]$ netstat -l -t -4 -n
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:20048           0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.10:4444       0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:17500           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:37222           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:875             0.0.0.0:*               LISTEN
```

Ilustración 31: Salida de consola del comando netstat

En el momento que el servidor recibe un comando del prototipo, lo procesa y realiza un reply con el comando “OK” de confirmación. En el procesamiento del comando recibido el servidor se encarga de actualizar el fichero web robotclean.php, que se encuentra en la raíz del servidor Apache, o si es un comando “alive” simplemente realiza el reply al dispositivo para confirmar al sistema de seguridad watchdog que sigue vivo.

En la siguiente ilustración se puede observar el diagrama de flujo del servidor TCP:

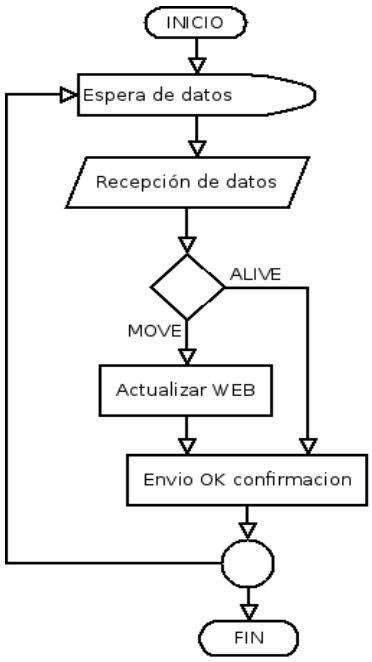

Ilustración 32: Diagrama de flujo del servidor TCP.

En la siguiente ilustración se muestra parte de la ejecución del servidor TCP:

```
RECEIVE: alive
SEND: OK
RECEIVE: MT:GO
SEND: OK
RECEIVE: alive
SEND: OK
RECEIVE: MT:BACK
SEND: OK
RECEIVE: MT:GIRO
SEND: OK
RECEIVE: alive
SEND: OK
RECEIVE: MT:GIRO
SEND: OK
RECEIVE: MT:GO
SEND: OK
```

Ilustración 33: Ejecución del servidor TCP

4.2.2 Interfaz de usuario.

La interfaz de usuario desarrollada en este proyecto es una web implementada, al igual que el servidor TCP, en el lenguaje de programación PHP. En ella se muestra el movimiento que esta realizando el robot en cada momento. Esta web está provista de un tiempo de refresco automático conilustraciónble a nivel de código.

El único requisito que requiere la interfaz de usuario, a parte de disponer del servidor TCP en ejecución, es disponer de un servidor Apache con soporte para PHP en el servidor remoto.

En la siguiente ilustración se muestra una captura de la interfaz de usuario en ejecución:

Ilustración 34: Ejecución de la interfaz de usuario.

4.3. Aplicación Principal.

La aplicación principal del sistema será la encargada de ejecutar todas las tareas con las funciones necesarias para conseguir los objetivos mínimos propuestos.

Este programa principal (llamado también “main.c”), estará creado como un proyecto FreeRTOS, el cual a su vez, se apoyará en varios proyectos de librerías estáticas.

Uno de los proyectos de librería estática es TFC_Library el cual será donde tendremos las librerías y/o drivers que hemos desarrollado a lo largo del proyecto y se han explicado anteriormente. Este proyecto ha sido desarrollado a lo largo de la realización del proyecto.

Otro proyecto de librería estática será “FreeRTOS_Library”, donde se encuentran las librerías de funciones propias del sistema operativo FreeRTOS. Dentro de este proyecto encontraremos las librerías necesarias para la utilización de este sistema operativo, como por ejemplo:

- task.h: Librería de obligado uso para la creación y uso de tareas en nuestro sistema.
- semphr.h: Librería de obligado uso para la creación de semáforos y funciones para trabajar con los mismos.
- queue.h: Librería de obligado uso para crear y trabajar con colas de comunicación entre tareas.

Una cola es una estructura de datos, caracterizada por una secuencia de elementos en la que la operación “push” se realiza por un extremo y la operación “pop” por el otro.

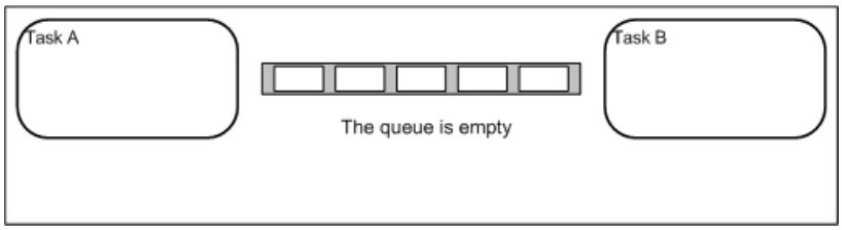

Ilustración 35: Representación de un acola en FreeRTOS

Existen varias formas de crear tareas, semáforos y colas bajo FreeRTOS, al igual que muchas formas de programar una aplicación principal.

Nuestra aplicación como se ha comentado en otros capítulos parte de la fase de formación, concretamente de un sistema productor/consumidor, adaptada a nuestro sistema, es decir, se han incluido nuestros dispositivos con sus librerías y utilizado las funciones correspondientes.

Para finalizar, en las siguientes ilustraciones podremos observar el diagrama de flujo de la aplicación principal, así como los diagramas de cada tarea:

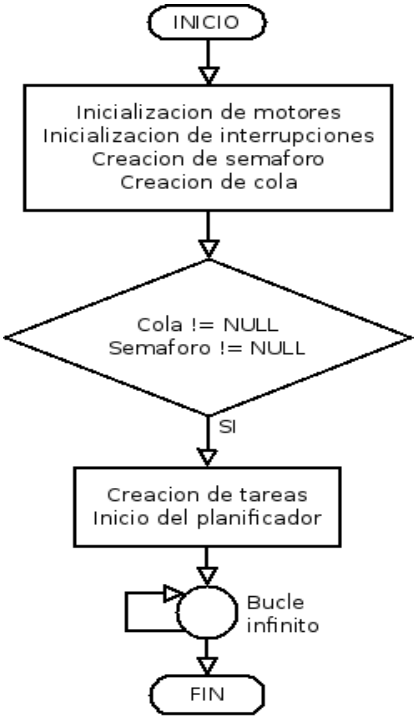

Ilustración 36: Diagrama de flujo main

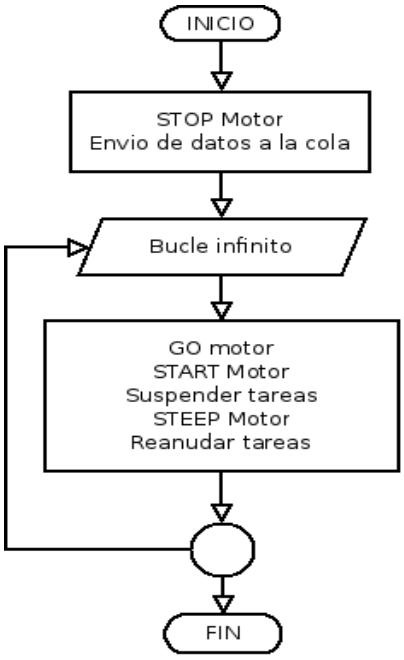

Ilustración 37: Diagrama de flujo vTaskMotorDefault

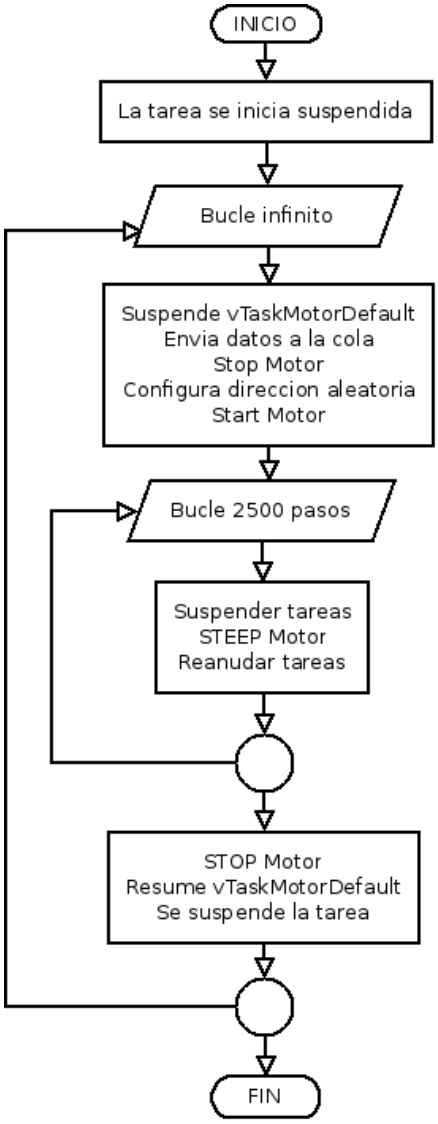

Ilustración 38: Diagrama de flujo vTaskMotorGiro

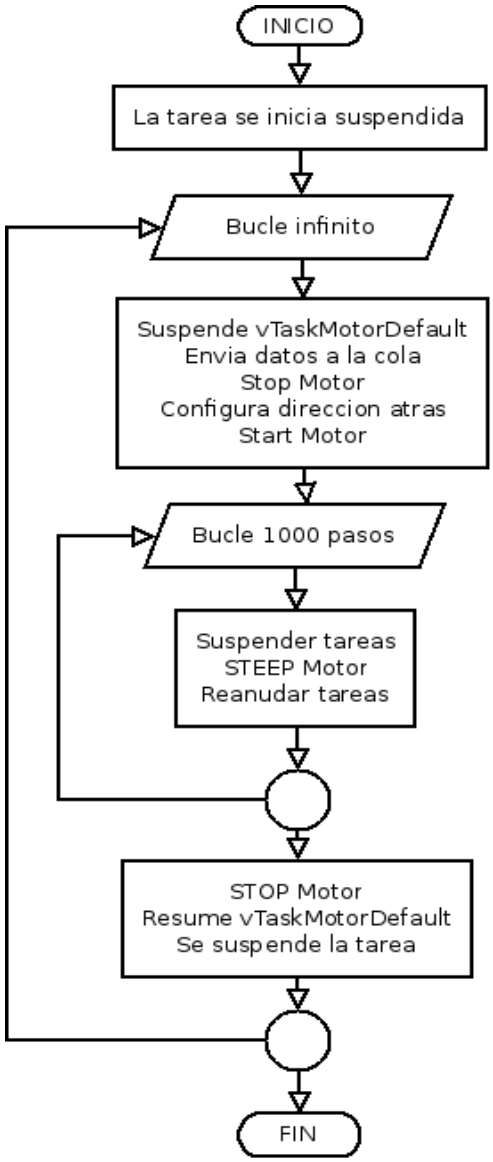

Ilustración 39: Diagrama de flujo vTaskMotorBack

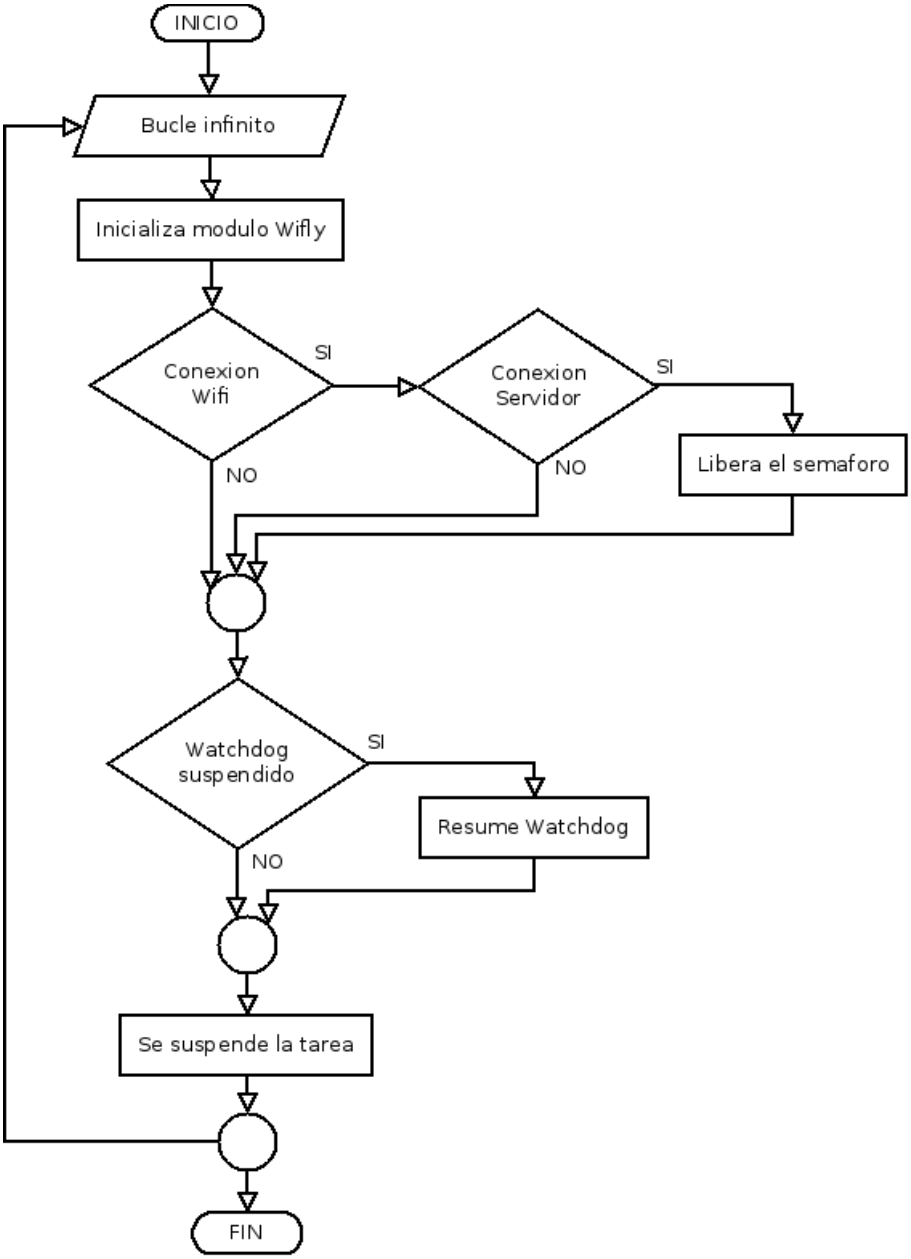

Ilustración 40: Diagrama de flujo vTaskConnection

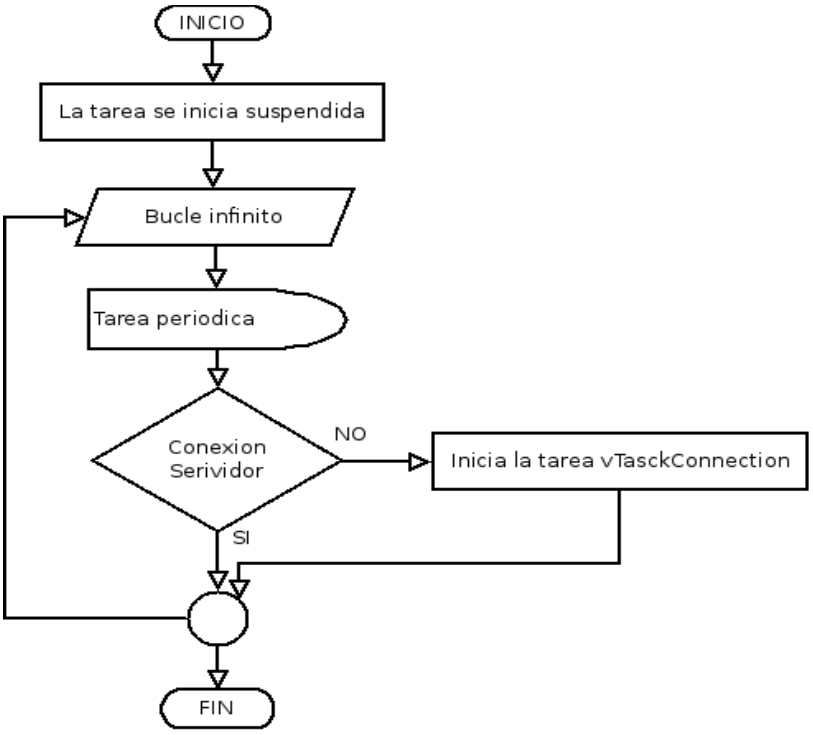

Ilustración 41: Diagrama de flujo vTaskWatchDogConnection

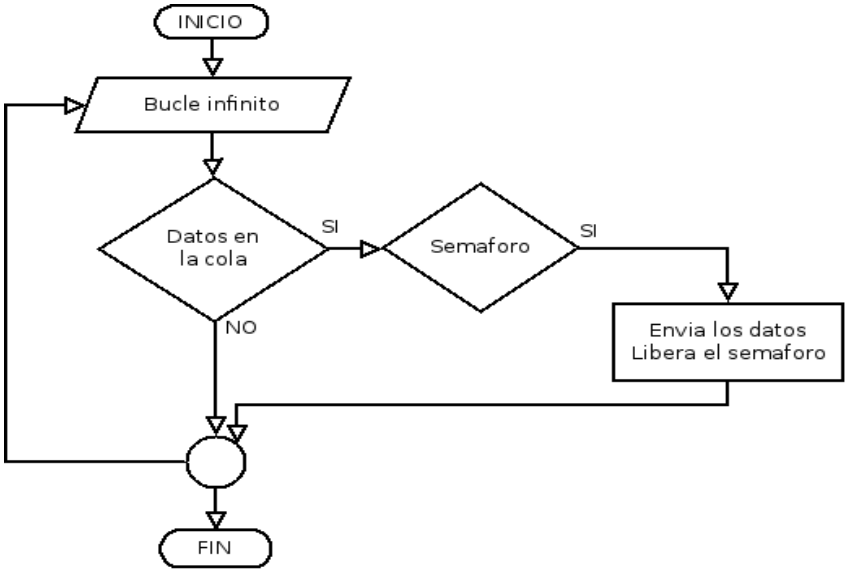

Ilustración 42: Diagrama de flujo vTaskSendData

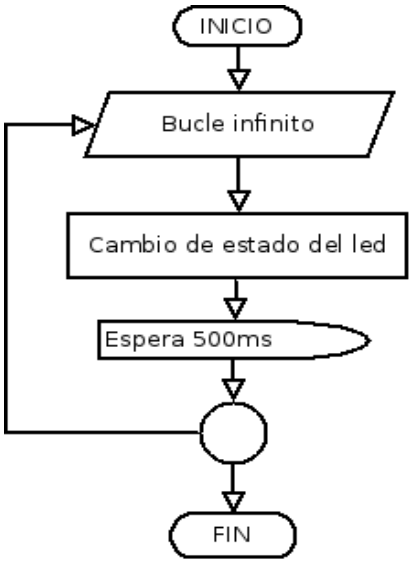

Ilustración 43: Diagrama de flujo vTaskRun

5. Viabilidad técnica.

El proyecto como puede comprobarse esta falto de madurez para poder ser considerado desde un punto de vista comercial totalmente viable, dado que no se han implementado los diseños de los pcb a nivel electrónico, ni los diseños de caja para su ensamblado, al igual que la programación de interfaces de usuario se han realizado de una forma simple a modo de demostración, debido al corto tiempo disponible para su realización.

Sin embargo, desde el punto de vista técnico, si puede considerarse un proyecto viable, ya que en la selección de sus componentes se ha tenido presente siempre la consecución de unos fines profesionales, teniendo en cuenta la calidad y precisión de los componentes. Por otro lado resuelve ampliamente el problema planteado, obteniendo un producto con capacidad para la movilidad autónoma por una estancia, envío de datos a un servidor remoto, así como capacidad para tomar decisiones de giro según los datos adquiridos de los sensores instalados dando por tanto solución al problema que se planteaba en este proyecto.

Listaremos a continuación los puntos fuertes y los puntos débiles más destacados:

Puntos fuertes:

- Flexibilidad funcional.
- Posibilidad de conexión de diversos sensores, mecanismos de control, etc...
- Adaptación a aplicativos web existentes.
- Ahorro de costes (por ejemplo licencias de software).
- Hardware bastante económico.
- Multitud de documentación y posibilidad de soporte de la comunidad Open Source.

Puntos débiles a mejorar:

- La autonomía energética y sistema de recarga autónoma de energía.
- Seguridad intrínseca del prototipo.
- Posibles errores y falta de depuración del código. Mejora de código en su conjunto.
- Peso del prototipo y amortización de la parte mecánica.
- Varios puntos posible de fallo (servidor web, router wifi de acceso, etc..)

Quiero remarcar finalmente que en este apartado se habla sólo de la vertiente técnica del proyecto. En lo que se refiere a la vertiente educativa, creo que el proyecto tiene mucho interés y cumple a la perfección su función.

6. Valoración económica.

Se detalla en la siguiente tabla la valoración económica de todo el desarrollo del proyecto, coste de material y tiempo:

Descripción Componentes	Cantidad	Precio Unidad €	Precio Total €
LPCXPresso LPC1769	1	14,04	14,04
CP2102	1	9,78	9,78
WiFly RN-171	1	24,65	24,65
Motor NEMA 17	2	17,3	34,6
Pololu A4988	2	9,95	19,9
AVR ATtiny85	1	2,75	2,75
HCSR04	1	15	15
Microinterruptor	1	1,55	1,55
Bateria Li-on 12v	1	18	18
Protoboard 75x50mm	1	3,32	3,32
Regulador L7905CV 5v	1	0,15	0,15
Regulador TS1117 3.3v	1	0,44	0,44
Diodo LED	1	0,3	0,3
Resistencia 10k	6	0,02	0,12
Cables varios	1	12	12
Total Coste Material			156,6
Total Coste Desarrollo en horas	80	20	1600
		Coste Total:	1.756,60 €

Ilustración 44: Presupuesto de desarrollo

Respecto a los costes que podría generar su utilización y recambios no se han contemplado ya que habría que contactar con un proveedor de recambios de mopa desechable para que presupuestara el coste de la fabricación a medida para nuestro prototipo.

7. Conclusiones.

Este TFC en su globalidad ha cumplido con expectativas de trabajar con muchas asignaturas estudiadas a lo largo de estos años.

En cuanto a sistemas embebidos se refiere, cumple con creces una primera toma de contacto, además de motivar al ingeniero a seguir trabajando en estos sistemas.

Se ha aprendido como integrar dispositivos nuevos, analizar protocolos de comunicación, capacidad de adaptación a las necesidades del cliente, crear funciones específicas, y sobro todo, llevar lo planificado a la realidad.

8. Glosario de términos.

ADC. Conversión analógica digital.
ARM. Arquitectura RISC (Reduced Instruction Set Computer) de 32 bits desarrollado por ARM Holdings.
Baudrate. Número de unidades de señal por segundo.
FreeRTOS. Free Real Time Operating System, es un sistema operativo en tiempo real para dispositivos embebidos.
GPIO. Entrada-salida de propósito general.
HTML. HypertText Markup Language, es un lenguaje de marcado para el desarrollo de páginas web.
Internet de las cosas. Término que se refiere a una red de objetos cotidianos interconectados.
Interrupción. Señal eléctrica asíncrona enviada por un periférico al procesador. Cada vez que se envía este tipo de señal, se interrumpe la ejecución normal de la tarea que este activa en el procesador en aquel momento. Cuando esto sucede, se deja en contexto actual (puntero de instrucciones, estados del registro, etc.) y se ejecuta una rutina del servicio de interrupción. Cuando finaliza la rutina de servicio de interrupción, el procesador continua con la ejecución normal de la tarea que había quedado interrumpida.
ISP. Interrupt service process, rutina de servicio de interrupción.
ISR. Rutina de servicio de interrupción, pequeño programa que se ejecuta en respuesta a una interrupción determinada. Es equivalente al termino gestor de interrupciones.
Microcontrolador. Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria.
Planificador o **scheduler**. Componente funcional que reparte el tiempo disponible de un microprocesador entre los procesos preparados para su ejecución.
Servidor. Nodo que provee servicios a nodos clientes.
Sistema empotrado. Sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.
RSSI. Indicador de fuerza de señal de recepción. Escala para medir el nivel de potencia de una señal inalámbrica.
UART. Transmisor/Receptor asíncrono universal. Controla los puertos y dispositivos serie.
Wi-Fi. Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

9. Bibliografía.

Libros y manuales.

- “Using the FreeRTOS Real Time Kernel - A Practical Guide NXP LPC17xx Edition” y “FreeRTOS Reference Manual” de Richard Barry.
- “Sistemas Empotrados en Tiempo Real” de José Daniel Muñoz Frias.
- “LPCXpresso User Guide” Rev. 7 — 18 February, 2014.
- UM10360 “LPC176x/5x User manual” Rev. 3 — 20 December 2013
- “User Manual and Command Reference” 802.11 b/g Wireless LAN Modules WIF LY GSX RN-131G, RN-131C, RN-134, RN-121, RN-123 & RN-125, RN-370 WIF LY EZX RN-171, RN-174, RN-XV Firmware Version 2.32 February 13, 2012.

Datasheets.

- LPC1769/68/67/66/65/64/63 32-bit ARM Cortex-M3 microcontroller; up to 512 kB flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN Rev. 9.3 — 8 January 2014 Product data sheet http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf
- LPCXpresso LPC1769 Rev B http://www.embeddedartists.com/sites/default/files/docs/schematics/LPCXpressoLPC1769revB.pdf
- RN-XV DataSheet RN-XV-DSv0.3 8/18/2011 http://www.rovingnetworks.com/files/resources/WiFly-RN-XV-DS.pdf
- A4988 Stepper Driver Carrier http://www.pololu.com/file/download/a4988_DMOS_microstepping_driver_with_translator.pdf?file_id=0J450
- Motor NEMA 17 http://www.pololu.com/file/download/SY42STH47-1206A.pdf?file_id=0J685
- Sensor HC-SR04. http://www.micropik.com/PDF/HCSR04.pdf

Recursos Web.

- Web de Sistemas Empotrados de la UOC. http://cv.uoc.edu/app/mediawiki14/wiki/IniciCortexM3
- Pagina oficial del sistema operativo FreeRTOS, http://www.freertos.org/
- Ejemplos de proyectos LPCXpresso. http://www.nxp.com/lpcxpresso-support
- Foro de LPCXpresso NXP. http://knowledgebase.nxp.com/
- Soporte Code Red. http://support.code-red-tech.com/CodeRedWiki/WikiHome
- Manual de PHP, http://php.net/manual/es/index.php
- Ansi C. http://c.conclase.net/librerias

10. Anexos.

10.1 Ejecución y compilación del código fuente.

10.1.1 Instalación del workspace.

Los pasos a seguir para la instalación del workspace que contiene el código fuente son los siguientes:

1. Importamos el “workspace” del proyecto pulsando sobre “import project”:

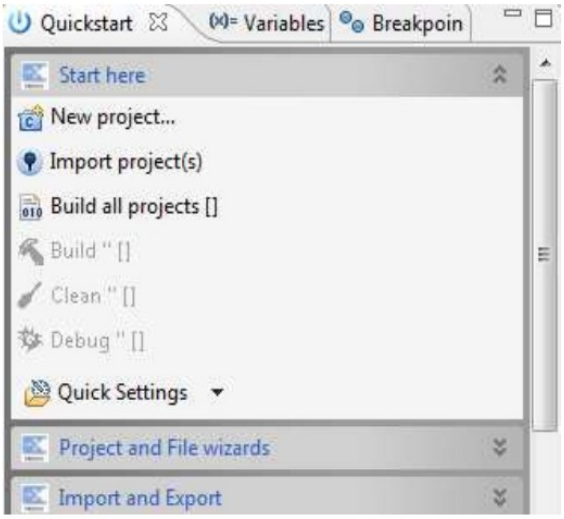

Ilustración 45: Instalación Cuadro de dialogo “import project”

2. Seleccionamos el archivo zip que contiene el código fuente:

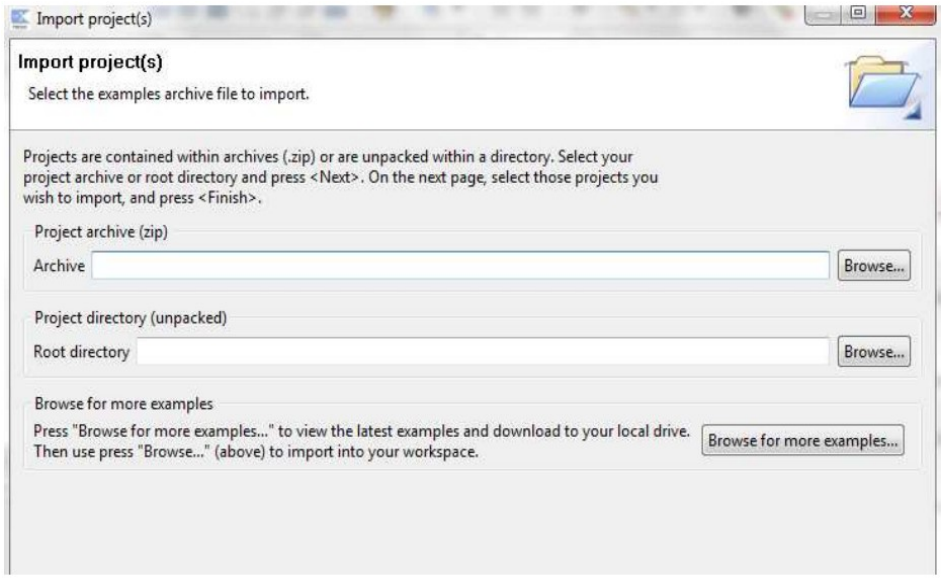

Ilustración 46: Instalación Cuadro de dialogo "select file to import"

3. Pulsando el botón finalizar del cuadro de dialogo anterior se finaliza la importación del workspace, quedando preparado para trabajar con él.

10.1.2 Compilación del código fuente.

Una vez tenemos el código fuente en nuestro workspace, procedemos a la compilación pulsando con el botón derecho sobre el proyecto TFC_Robotclean y seleccionando "Build Project":

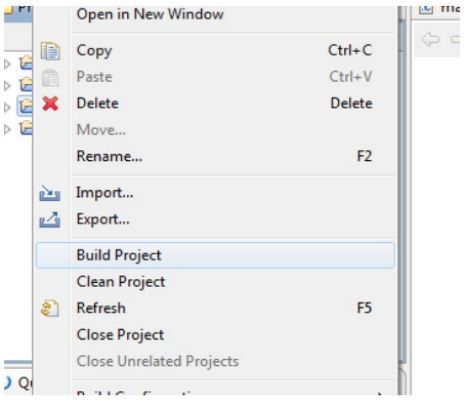

Ilustración 47: Compilación del proyecto TFC_Robotclean

10.1.3 Ejecución del proyecto.

Para ejecutar el código, procediendo de la misma forma que en la compilación, pero esta vez seleccionaremos “Run As -> C/C++ MCU Application”:

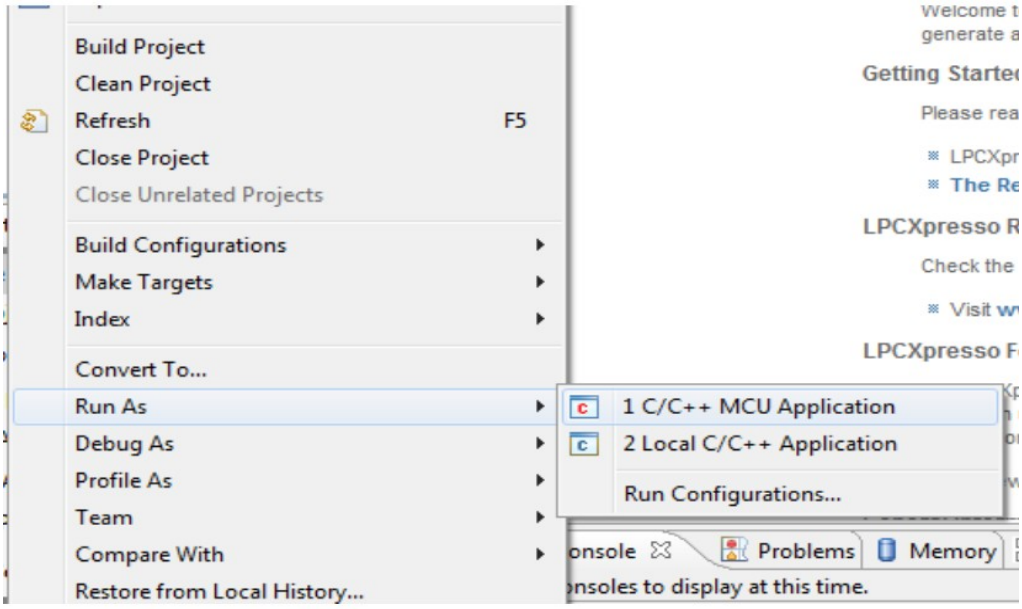

Ilustración 48: Ejecución del proyecto

10.2 Instalación y ejecución del servidor TCP y la interfaz de usuario.

El desarrollo de este proyecto se ha realizado sobre una máquina con un sistema operativo GNU/Linux, por este echo las pruebas de ejecución que se muestran a continuación son sobre esta plataforma. Como el desarrollo se ha realizado sobre el lenguaje de programación PHP, este programa es portable a otras plataformas, sin prácticamente ninguna modificación a nivel de código.

10.2.1 Servidor TCP.

El servidor TCP no requiere ninguna instalación, solo es necesario que a nivel de host tenga soporte para el lenguaje de programación PHP.

Como se muestra en la siguiente ilustración, para la ejecución del servidor simplemente hay que ejecutar el script con permisos de superusuario:

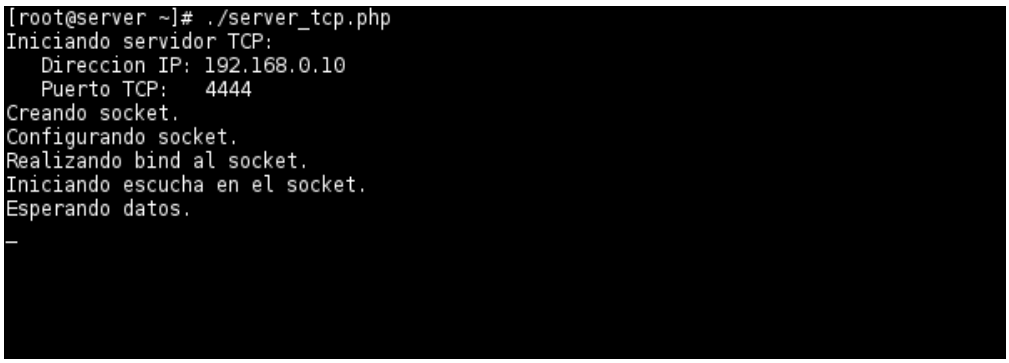

Ilustración 49: Instalación y ejecución del servidor TCP.

10.2.2 Interfaz de usuario.

La interfaz de usuario esta formada por una pagina web alojada en un servidor Apache con soporte para PHP. La instalación de la interfaz de usuario web se realiza como podemos ver en la siguiente ilustración:

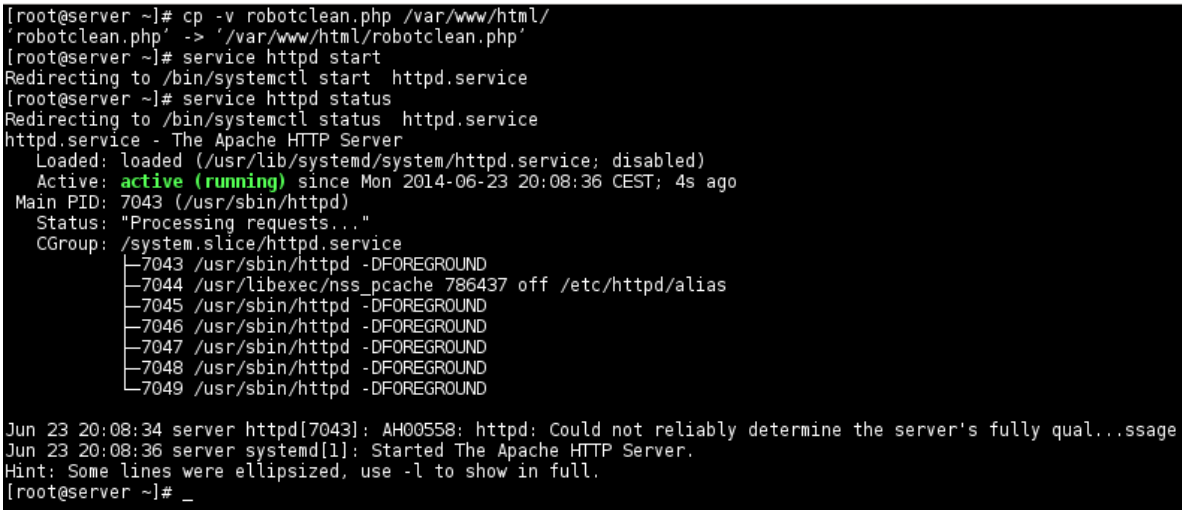

Ilustración 50: Instalación de la interfaz de usuario web.

La ejecución de la interfaz de usuario se realiza, como podemos ver en la siguiente ilustración, accediendo a través de un navegador web la dirección web con el formato “http://<direccion_ip_del_host>/robotclean.php”:

Ilustración 51: Ejecución de la interfaz de usuario.